パソコンプログラマのためのクリエイティブマガジン

1991年3月1日発行(毎月1回発行)　第1巻　第3号　通巻3号

月刊プログラマーズ・ページ

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

1991 3 MARCH

定価980円

特集 この言語を使いなさい!!

数ある言語の中から主要14言語をピックアップ
それぞれの言語を使い込んだプログラマが
その特性から使い方のポイントまで解説

創刊記念プレゼント
HANDY98他

SE
Shoeisha Co., Ltd.

Microsoft

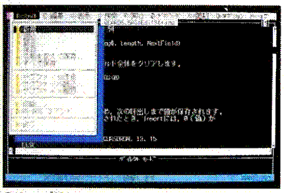
使いやすい統合環境と高度で多彩な機能。Quickシリーズは、プログラミン

Microsoft QuickBASIC Ver. 4.5は、アプリケーション感覚の自由自在な操作性とパワフルな言語仕様をもった最新のBASIC言語。Quickシリーズならではの使いやすい環境とプロユースにも耐える高度な機能を搭載。他のBASICとは根本的に異なる新世代言語です。

もちろん、初めて言語を学ぶ方もプログラミング言語の基本学習ツールとして活用していただけます。また、構造化とモジュール化の強力なサポートにより、小回りのきくユーティリティの作成から高度なアプリケーションの開発まで、プロフェッショナルの方にも、即戦力としてご利用いただけます。

アプリケーション感覚で使える統合環境

Quick BASICの統合環境はエディタ、インタープリタ、コンパイラ、デバッガをすべて標準装備。エディタのドロップダウンメニューから、全ユーティリティを実行できます。基本メニューは、利用頻度の高いコマンドだけを集めたEasyメニューと、全機能を網羅した高機能なFullメニューの2種類を提供。また、プログラム入力以外のすべての操作はマウスでOK。エディタを高速化し、カーソル惰性も抑止しました。



●ドロップダウンメニュー

入力ごとに構文チェック

プログラムを一行入力するごとに構文チェック。文法的な誤りなどに対しては、即座にエラーメッセージを表示します。イージーミスがすぐに発見でき、さらにスムーズなプログラム開発を実現します。

モジュール化の概念を導入

サブルーチン、ユーザー関数の考え方を大幅に拡張。プロシージャの概念を導入し、プログラムをモジュール単位で管理。1度作ったサブモジュールは、他のプログラムに再利用できます。また、MS-CやMASM、MS-FORTRAN、QuickCで作成したプロシージャを呼び出し、関数やステートメントのように利用することもできます。

＊Microsoft C 6.0には対応していません。

構造化プログラミングをサポート

構造化のための制御文をフルサポートし、構造化プログラミングを強力にバックアップ。BASICを行番号から解放し、スパゲティ・プログラムを追放します。

初心者にやさしい学習環境

QBチュートリアル：QuickBASICの画面上の名称や構造化言語・モジュール化の概念などを学べます。マニュアルにはプログラム例をまじえた解説やプログラム作成の工程が分かりやすく説明されています。

QuickBASICプログラミングガイド：サブプログラムの作り方やデバッグの方法などプログラミングの基本的な流れが、サンプルプログラムを完成させる形で学べます。

QBアドバイザ：オンラインヘルプ機能を飛躍的に進化させた強力なQBアドバイザが、統合環境の操作方法、各関数やステートメントの詳細な説明、プログラム例、エラーメッセージの解説、操作方法の説明などをスピーディーに表示します。いちいちマニュアルを読む必要はありません。

●QBアドバイザ

強化・拡張されたステートメントと関数

PLAY*、SOUND*、SLEEP、UEVENTステートメントを追加。また、PRINT、LOCATE、CIRCLEステートメントの処理を高速化。Ver. 4.2のステートメントや関数も強力な言語仕様へ拡張。実行ファイルの処理速度は非常に高速です。

＊PC-9801E/F/M/Uでは利用できません。また、PLAYステートメントはFM音源ボードには対応していません。

多彩なデバッグ機能

ブレークポイントを設定して実行、変数の内容を見ながらトレース、一定条件を満たすまで実行など、多彩なデバッグ機能を備えています。

高機能インタープリタと高速コンパイラ

編集画面と実行画面を独立して管理する統合環境を持ったインタープリタと、Quick-BASICがなくてもMS-DOS上で動作する高速なEXEファイルを作成するコンパイラを標準で装備。いずれも数値演算コプロセッサに対応しています。

パス設定コマンドの追加

メタコマンドで読み込まれるインクルードファイルやライブラリ、実行ファイルなどのパス設定が行えます。

実用的なライブラリ

マウス対応のプログラムを作るためのプロシージャ群や、漢字入力システムのオン/オフ制御、グラフィック画面のスクロールやグラフィック画面への文字描画、画面ハードコピールーチンなど、実用的なライブラリを用意。ソースプログラムも付属しています。

■基本システム

〔NEC PC-9800シリーズ〕

- 本体NEC PC-9800シリーズ(ただし、PC-9801、PC-98XA/LTおよびXL/XL^2/RLのハイレゾリューションモードを除く)
- 日本語MS-DOS Ver. 2.0以上
- 必要メモリ：640Kバイト
- 利用可能な日本語システム：MS-KANJI API仕様のもの。もしくはATOK6。
- ハードディスク、マウスのご利用をお勧めします。

Quickが

Microsoft® QuickBASIC

Version 4.5

NEC PC-9800シリーズ対応　標準価格20,000円(消費税別)　好評発売中

グ言語の学習から本格的アプリケーション開発まで、強力にサポートします。

パソコン用プログラム開発の主力言語として、いま最も人気の高いC言語。しかし、多数のユーティリティが必要、いちいちマニュアルを読まなくては使いこなせない、動作確認の度に実行ファイルが必要、論理的な誤りをおかしがちな文法、膨大な関数を覚えなくてはならないなどの点で、初心者には難しいといわれているのも事実です。
でも、QuickC2.0なら大丈夫。これらの問題点を解決し、初心者の方にもやさしいプログラミング環境を提供します。しかも、従来バージョンに比べて機能は飛躍的にアップ。C言語の学習からプロユースまで、最良のCコンパイラとしてご活用いただけます。

プログラム開発を容易にする統合環境

エディタ、コンパイラ、リンカ、デバッガを統合環境上で利用可能です。起動するとエディタの状態になり、すぐにプログラムが書けます。書き終わったら、後はファンクションキーを押すだけ。コンパイル&リンク&実行は自動的に行われます。もちろん実行ファイルも自動作成します。

初心者にやさしい学習環境

QCアドバイザ：オンラインヘルプ機能を強化改良したQCアドバイザが、場面に応じて適切にアドバイス。操作方法、ランタイム関数やキーワードに関する解説が、編集しながら呼び出せます。しかも、コンパイル中に発生したエラーの説明もすぐに参照できます。また、ランタイムライブラリ関数の説明には、すべてサンプルプログラムが付いています。さらに、付属のHELPMAKEユーティリティを利用して、標準のヘルプファイルを作り直したり、オリジナルのヘルプファイルを作成することも可能です。

●QCアドバイザ

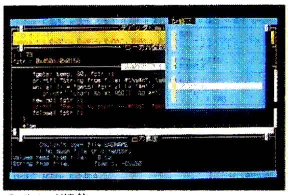
CBT：対話形式のComputer Based Training(CBT)で、統合環境の使い方、プログラムの作成、デバッグの方法などをコンピュータ上でスピーディーに学習できます。

●CBT

C for Yourself：初心者の方のための学習マニュアル「C for Yourself」が付属。難しいC言語の文法を、豊富なサンプルプログラムをもとに分かりやすく解説してあります。しかも、サンプルプログラムは、QCアドバイザから簡単に呼び出せます。

強化されたデバッグ機能

Quick C 2.0は、強力なデバッグ機能を内蔵します。ローカルウォッチ変数や実行画面をウィンドウで表示できます。さらに、レジスタウィンドウもサポートします。また、Quick C 2.0のデバッグ機能は、同一の環境上で実現されるので、作成したプログラムをコンパイル&リンク後、そのソースプログラムを表示したままトレースやブレークポイントの設定をすることが可能です。



●デバッグ機能

やさしいインタフェース

メニュー選択や編集操作などほとんどの操作がマウスで行えます。また、編集キーでの編集方法は、Microsoft WindowsやMicrosoft Worksと共通なので、とても簡単。覚えた操作はこれからのマイクロソフト製品でもそのまま利用できます。

豊富なユーザーカスタマイズ機能

編集キーの割り当てを変更するMKKEYユーティリティ、画面の表示色を自由に変更できるQCCONFIGユーティリティを追加。さらに、ユーザーによるメニューの追加を実現するカスタマイズメニュー機能も装備されます。

充実したライブラリ

実数座標をサポートした"グラフィックスライブラリ"、グラフやチャートが簡単に書け、多種類のフォントが揃った"チャート/フォントライブラリ"、BIOSコールを直接サポートする"BIOSライブラリ"、MS-KANJI API仕様の日本語フロントエンドプロセッサをアプリケーションから制御する"カナ漢サポートライブラリ"を追加しました。

NMAKEユーティリティ

UNIX上のMAKEユーティリティに近い機能を持つ最新のNMAKEユーティリティを標準装備しています。統合環境上でNMAKEが利用可能なMAKEファイルも自動作成することができます。

インクリメンタルコンパイル&リンクを実現

コンパイル&リンクがさらに高速に。しかも統合環境上でコンパイル可能なソースファイルのサイズも飛躍的に拡大しました。

基本です。

■基本システム
〔NEC PC-9800シリーズ〕
●本体NEC PC-9800シリーズ(ただし、XA・LTおよびXL・XL^2・RL・H98のハイレゾリューションモードを除く)
●日本語MS-DOS Ver2.0以上
●必要メモリ：640Kバイト
●利用可能な日本語システム：MS-KANJI API仕様のもの。もしくはATOK6。
●ハードディスク、マウスのご利用をお勧めします。

Microsoft® QuickC®

Compiler Version 2.0

NEC PC-9800シリーズ対応　標準価格20,000円(消費税別)　好評発売中

ATOKは、㈱ジャストシステムの登録商標です。Microsoft、MS、MS-DOS、QuickCは、米国マイクロソフト社の登録商標です。
MS-KANJI APIは、マイクロソフト株式会社が推奨する日本語入力システムの仕様です。

マイクロソフト株式会社
〒160 東京都新宿区西新宿7-5-25 Kビルディング　電話(03)3363-1201

〈資料請求番号P02〉

SE 翔泳社が贈る

QuickBASIC スタイルブック

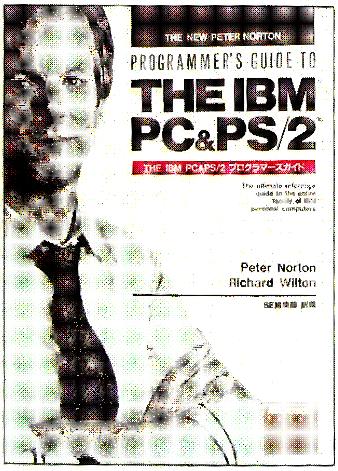
●定価2,300円

イント・ツーワン著

QBを骨まで しゃぶってみませんか

数あるプログラミング言語の中でも、Quick BASICは、初心者からマニアまで幅広く使われています。この本は、少しでもわかりやすく、質の高いプログラミングを書くために一問一答式の問題集形式になっている「良いプログラミング」のための学習書です。さあ、あなたも挑戦してみてはいかがですか。

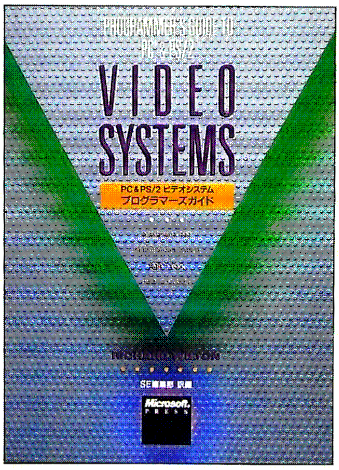
IBM PC&PS/2プログラミング3部作



THE IBM PC&PS/2 プログラマーズガイド

●Peter Norton
& Richard Wilton 著
●定価5,200円

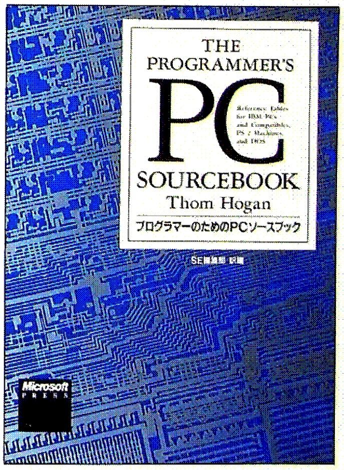
あのピーター・ノートンのベストセラー。IBMプログラマ必携のスタンダードガイダンス。



PC & PS/2 ビデオシステム プログラマーズガイド

●Richard Wilton 著
●定価6,000円

IBM PCファミリーのビデオシステム完全ガイド。



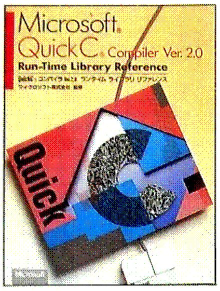
プログラマーのための PCソースブック

●Thom Hogan 著
●定価5,200円

IBM PC,PS/2プログラマ必携の技術資料完訳。

プログラミングブックス

マイクロソフトQuickシリーズのスペシャルリファレンス



QuickC コンパイラVer.2,0 ランタイムライブラリリファレンス

QuickBASIC Ver. 4,5 ステートメント&関数リファレンス

●マイクロソフト株式会社 監修 ●定価各2,400円

マイクロソフトの超人気言語ソフトQuickシリーズ。Quick BASIC Ver. 4, 5 Quik C Ver. 2,0スダンダードリファレンス。マイクロソフト社監修でオンラインヘルプを完全書籍化サンプルプログラム満載のリファレンス。

初級〜中級プログラマーのためのガイダンス

Turbo C Ver. 2,0 プログラミングブック

●田中正造 著 ●定価2,000円

BGI／ROM BIOS/EMSの実践テクニックガイダンス、Turbo C 2,0の実徴的な機能を豊富な図表と実践有なサンプルプログラムで解説。

QuickBASIC Ver. 4,5 プログラミングブック

●塚本吉彦 著 ●定価2,400円

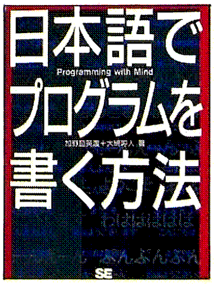
Quick BASICで気軽にプログラミング。BASICビギナーだけではなく昔からのBASICファンにも贈るQuick BASICの入門書です。

日本語プログラミングへの招待

日本語プログラミング言語 MIND

●片桐明 著 ●定価2,884円

日本語でプログラムが書ける画期的なソフト「Mind」。開発者自身の執筆による決定版解説書。



日本語でプログラムを書く方法

●加野島英渡・大網呼人 著

●定価1,200円

本書には、記号、数式は一切出てきません。でもプログラミングの本なのです。日本語で書いたプログラムが本当にパソコンで動きます。

ディスク付MS-Cリファレンス

MS−C Ver. 5,1 オンラインリファレンス

●谷口崇 著 ●定価2,200円

フロッピーディスク付のプログラミングツール。

エッセイ

❖読書するプログラマ ●祐安重夫 著 ●定価1,648円

「インターフェース」誌連載。博学でならす著者による読書日記。

❖デジタル版徒然草 ●園田康篤 著 ●定価1,545円

「PCワールド」誌連載。一流のプログラマが世相を斬る。

※定価はすべて消費税込みとなっております。

SE
株式会社 翔泳社
東京都千代田区平河町2-4-14 平河町KSビル 〒102
PHONE 03-3263-0447 FAX 03-3263-4526

〈資料請求番号001〉

Brief

Full Screen Editor 日本語版

全米のプログラマが認めた実力！
アメリカでは「ブリーフ」が
プログラミング用エディターとして
最も多く利用されています。

C言語で組めるマクロ！

Version UP!

v.3.0 いよいよ登場。

プログラミングエディター新世紀。

よりパワフルで、よりエレガントな
プログラミングのために……。

●C言語で組めるマクロ 新機能

使い慣れているC言語でマクロが組めますから、習得が簡単でカスタマイズも思いのままです。

●マクロのスクリーン・デバッガ 新機能

ソースコードでデバックできます。もちろんブレークポイントやウォッチ・ドッグもセットできます。

●テンプレート機能〈構文の自動生成〉 新機能

C、Ada、Basic、COBOL、FORTRAN、MODULA2、Pascalで構文を自動生成します。

●300回ものアンドゥ

各ファイルごとに300までの操作を元にもどすことができますので、やり直しの面倒を気にせず、安心して編集できます。

●安全編集

Briefは、一定時間キーを操作しないこと最新の編集内容をファイルに自動保存します。だから編集中に電源が切れても安心です。

●その他の主な新機能

キー操作のファイルへのセーブとリプレイ、処理可能行数を拡張、リジューム機能、Ctrlキー・アサイン(オプション)、メニュー(オプション)etc。

アメリカでも日本でも高い評価

●プログラミング向きエディターを探している人は、もう探す必要はない。(Jerry Pournelle/BYTE)
●ソフト開発には強力なツール、300回のアンドゥは大きな特徴。(日経バイト)
●既存のエディターに不満を覚えたら、これしかない。(Oh! FM)
●まるで夢のようなアンドゥだ――おもわず間違うのを期待してしまいそうだ。(The C Users Journal)

■主な機能

◆55ウィンドウ◆無限のファイル◆自動コンパイル◆自動エラー・ジャンプ◆自動インデント◆自動テンプレート◆バッファ書出/切換/削除/リスト表示◆ファイル名の指定/表示/変更◆バックアップ保存◆特殊文字入力◆2行を1行に連結◆ファイル単位の挿入◆増分検索◆各種マーク/始・終点の入換◆ブロックのカット/コピー/ペースト/印刷/保存/左右へのシフト◆対応する括弧へ移動◆C言語のコメント化/非コメント化◆DOSコマンドの実行◆キー操作記憶/実行◆入力ヒストリー◆英文テキスト整形◆タブのスペース挿入◆拡張子毎に異なった設定ができるタブ◆検索置換パターンの画面からの入力◆英字の大小文字変換 その他

定価 48,000円

(消費税は含まず)

■5機種対応パッケージ

(東芝、NEC、AX協議会、IBM、富士通)

●5'2DD、3.5'2DDデュアルメディア●MS-DOS 3.1以降に対応●英語DOSでは使用できませんが、英語版BRIEFもセット販売いたしております。●お急ぎの方、上記以外のメディア希望の方、英語BRIEFのセット販売希望の方はユーザー・サポート係までご連絡下さい。

日本システム株式会社

●お問合せは　東京都立川市曙町1-18-2一清ビル立川事業所Briefユーザー・サポート係
TEL0425-23-4469　FAX0425-27-6127

〈資料請求番号002〉

Programmer's Page
MONTHLY MAGAZINE FOR PROGRAMMERS

パソコンプログラマのための
クリエイティブマガジン
月刊プログラマーズ・ページ

1991 MARCH No.3

CONTENTS

特集
この言語を使いなさい!!

パソコン・プログラミング言語盛衰記 11

プログラミング言語の仲間たち 神林靖 18
その誕生から系譜を辿る

私が勧めるこの言語

Assembler
TSR、デバドラなどに使われるDOSの主力言語 小高輝真 28

BASIC
すぐに乗れる自転車のような手軽さ 山本英昭 33

C
考えたことを素直に記述できる 辻村浩 37

C++
オブジェクト指向を取り入れた次世代C言語 大野一久 43

COBOL
大型機の経験を生かせる互換機 海老原浩之 46

FORTRAN
科学技術計算の世界共通語 桜井健次 50

HyperCard/HyperTalk
高度なソフトを容易に制作できる 田村浩二 55

Lisp
高速・大容量化時代に能力を発揮する 藤沢滋 58

LOGO
効率より学習の容易さを優先 福田洋 63

Mind
日本語プログラミング言語 加野島英渡 67

Modula-2
モジュール化を重視した文法 西端芳彦 71

Pascal
ミスを適切に指摘してくれる優しさ オケゲム 76

Prolog
強力なプログラムを速く記述する 鈴木太朗 80

Smalltalk
柔軟性のあるオブジェクト指向言語 近江武一 84

表紙イラスト:日暮修一
表紙AD:佐々木眞夫(SE)

Illustration: NAOKI SATO

C O N T E N T S

INFORMATION PAGE1 1
NEWS & NEW PRODUCTS 2
佐々木哲のニューヨークシステム通信 4
西川和久の希望的観測 6
Programmer's Rest Bench 8

パソコン博物館 第3回
ゲームマシンの先駆けとなった電子帳簿/コモドールPET2001 井戸禎光 9

プログラマの道具学 第3回
モデム 祐安重夫 92

実践C塾 第3回
MS-CとTurbo Cのコード生成の特徴 水居 浄 94
98活用アセンブラ技法/ビデオシステム編 第3回
フォント作成プログラム 星野操 104
プログラマのためのハードウェア再入門 3
セントロニクス 鎌田正寿 114
プログラム「遊」工房 参乃巻
迷路作成プログラム 谷口典生 118

今月の受験情報
自ら動機づけをして、まず2種から挑戦 赤松高文 125

INFORMATION PAGE2 129
BookReview「The Annotated C++ Reference Manual」中村正三郎130
本のアンテナ 古田島義和
Programmer's Program IOS-10STD 132
プログラマ登場③ 兵藤嘉彦 133
オンラインソフト採集帳3「WTERM Ver.8.20」134
HEALTHY? UNHEALTHY?
コンピュータの法律 熊谷信太郎 136
アキバ短信
バーチャルスペースガイド 138
創刊記念プレゼント 140
ドラゴン・ポスト
クイズ・ランダムスクリーン 142

付録
Microsoft C 6.0関連プログラム起動オプション

広告索引	資料請求番号
マイクロソフト(株)	P02
日本システム(株)	002
(株)ランハード	003(128ページ)
丸文(株)	P03
三菱電機(株)	P04

人材募集広告
マイクロソフト(株)
(掲載順)

information・page 1

●イベント

■2月27日(日)～3月1日(金)
ボード・コンピュータ'91
CPU、メモリ、周辺機能ボードなどの産業用コンピュータボードを一堂に展示。'91スイッチング電源システム展、EMC・ノイズ対策技術展、'91小形モータ技術展も同時開催。
会場：池袋サンシャインシティコンベンションセンターTOKYO／入場料：1,000円
問い合わせ先：(社)日本能率協会　TEL 03-3434-1377

■3月12日(火)～3月14日(木)
情報処理学会第42回全国大会
講演に加え、UNIXの将来と課題と題したパネルディスカッションも行われる。
会場：東京工科大学　参加費：1,000円(会員)、2,000円(非会員)、学生無料
問い合わせ先：(社)情報処理学会全国大会係　TEL 03-3505-0505

■3月18日(月)～3月20日(水)
1991年電子ディスプレイ展
国内外の電子ディスプレイデバイス、ディスプレイ応用製品を展示。
会場：科学技術館　入場料：無料
問い合わせ先：日本エレクトロニクスショー協会　TEL 03-3284-1051

■3月22日(金)～3月26日(火)
'91ストア・オートメーション・ショー
最先端の流通情報システム、SAシステムを一堂に展示。'91カード・ビジネス・フェアも同時開催。
会場：東京・晴海見本市会場　入場料：1,500円
問い合わせ先：日本経済新聞社事業局総合事業部　TEL 03-3252-8157

■3月26日(火)～3月28日(木)
1991年度精密工学会春季大会学術講演会
講演、パネル形式によるポスターセッションが行われる。
会場：早稲田大学理工学部　参加費：3,000円(会員)、5,000円(非会員)
問い合わせ先：(社)精密工学会　TEL 03-3362-4030

■3月26日(火)～3月29日(金)
1991年電子情報通信学会春季全国大会
一般講演、シンポジウム講演が行われる。
会場：徳島大学常三島キャンパス　参加費：1,000円(会員)、1,700円(非会員)
問い合わせ先：(社)電子情報通信学会　TEL 03-3433-6691

●海外イベント

■3月13日(水)～3月20日(水)
ハノーバー・メッセ　セビット'91
世界の事務・情報・通信技術を展示する。
会場：ドイツ・ハノーバー市営常設見本市会場
問い合わせ先：ドイツ産業見本市日本代表部　TEL 03-3349-3446

■3月18日(月)～3月20日(水)
The Sixth International Conference & Exposition on CD-ROM and Multimedia
CD-ROM、マルチメディアに関する展示と会議が行われる。
会場：米国・San Jose Convention Center
問い合わせ先：カーナーズエクスポジションジャパン㈱　TEL 03-3349-8501

●セミナー

Microsoft University

■2月25日(月)～3月1日(金)
Microsoft LAN Manager Applications Programming
LANマネージャ用アプリケーション開発のための上級コース。
会場：富士ソフトウェアMUセミナールーム
受講料：180,000円(5日間)

■2月26日(火)～3月1日(金)
MS OS/2 Programming Environment
OS/2アプリケーション開発のための基本コース。
会場：インテルジャパン東京トレーニングセンターMUセミナールーム
受講料：120,000円(4日間)

■3月4日(月)～3月8日(金)
MS OS/2 Applications Programming
OS/2のAPIを使った実習中心のコース。
会場：富士ソフトウェアMUセミナールーム
受講料：150,000円(5日間)

MS OS/2 Presentation Manager Programming Environment
OS/2 PMアプリケーション開発のための入門コース。
会場：インテルジャパン東京トレーニングセンターMUセミナールーム
受講料：150,000円(5日間)

■3月11日(月)～3月15日(金)
MS OS/2 Presentation Manager Applications Programming
OS/2 PMアプリケーション開発のための上級コース。
会場：インテルジャパン東京トレーニングセンターMUセミナールーム
受講料：150,000円(5日間)

Microsoft Windows Programming Environment
Windowsアプリケーション開発のための入門コース。
会場：富士ソフトウェアMUセミナールーム
受講料：150,000円(6日間)

■3月26日(火)～3月29日(金)
Microsoft LAN Manager Programming Environment
LANマネージャアプリケーション開発のための入門コース。
会場：富士ソフトウェアMUセミナールーム
受講料：150,000円(4日間)

■3月13日(水)～3月14日(木)
Microsoft C入門
実習をまじえてC言語の基本を解説。
会場：JRパソコンステーション上野セミナールーム
受講料：55,000円(2日間)

Microsoft Universityは、OS/2、Windowsアプリケーション開発者のために定期的に開催されているセミナーです。
問い合わせ先：富士ソフトウェアMU教育センター　TEL 03-3452-7722

■2月26日(火)～2月27日(水)
Macintoshの学校「マルチメディア実践トレーニング」
レーザーディスクとハイパーカードを使ったマルチメディアトレーニング
会場：池袋西武百貨店SMA館
受講料：109,180円(2日間)テキスト代・機材使用料込
問い合わせ先：西武百貨店IDFプロジェクト室Macintoshの学校
TEL 03-5992-8890

NEWS & NEW PRODUCTS

マイクロソフトウェア，Zortech C++の独占契約を締結．286版はWin 3.0にも対応

マイクロソフトウェア㈱(TEL 03-3818-7888)は，英国Zortech社と「Zortech C++コンパイラ V/2.1」の国内での独占販売，J-3100，PC-9800などの国産機種への移植に関しての契約を締結し，販売を開始した．Zortech C++は，AT&Tの最新仕様であるV/2.1に基づいた世界初のネイティブC++コンパイラで，MS-DOS，MS-Windows，OS/2用の「Zortech C++/286コンパイラ」(168,000円)および「同386コンパイラ」(280,000円)がある．

MS-Windows 3.0日本語版がリリース

マイクロソフト（株）は，「Microsoft Windows 3.0日本語版」を国内のパーソナルコンピュータメーカーにリリースした．製品化の第1弾は，日本電気（株）から発売されるPC-9800用「日本語MS-WINDOWS Ver.3.0」で，2月中旬から出荷を開始する．価格は25,000円．「日本語MS-WINDOWSソフトウェア開発支援キット」(90,000円)は，2月下旬から出荷される．

コンテック，自社ボードをサポートしたQB用ドライバソフト発売

㈱コンテック(営業部販売促進課TEL 03-3769-1061)は，自社製NEC PC-9800シリーズ用シリアル通信ボード「SIO-2(98)」シリーズをサポートしたQuickBASIC用ドライバソフト「SIO-PAC(98)QB」を発売した．SIO-PAC(98)QBは，SIO-2(98)シリーズを最大2枚，4チャネルまでサポートし，NEC純正RS-232Cボードとの同時使用，X-ON/OFFによるフロー制御も可能．価格は，16,000円．

日本電気，386を搭載したデスクトップ機など4機種16モデルを発売

日本電気(株)，日本電気ホームエレクトロニクス(株)(NECパソコンインフォメーションセンター：TEL 03-3452-8000)は，CPUに80386を搭載したPC-9800DA，80386SXを搭載した普及機DS，V30HLを搭載したUR，UFなどの新シリーズ4機種16モデルのデスクトップ機を発売した．

■PC-9801DA2/DA5/DA7

プロセッサ：80386（20/16MHz）／外部記憶装置：5インチFDD×2（DA2）・40MB HDD(DA5)，100MB HDD（DA7）／価格：448,000円(DA2)，598,000円（DA5），698,000円（DA7）

■PC-9801DS2/DS5

プロセッサ：80386SX（16MHz）／外部記憶装置：5インチFDD×2(DS2)・40MB HDD(DS5)／価格：358,000円（DS2），508,000円（DS5）

■PC-9801UF／UR／UR20

プロセッサ：V30HL（16/8MHz）／外部記憶装置：3.5インチFDD×2（UF)，×1・1.2MB RAM(UR/UR20)・20MB HDD（UR20）／価格：218,000円（UF／UR），348,000円（UR20）

マイクロソフト，Windows 3.0対応のExcelを発売

マイクロソフト㈱（TEL 03-3363-1201）は，Microsoft Excelをアップデートし，MS-Windows 3.0の日本語版に対応した「Microsoft Excel Ver.2.1 for Windows 3.0」を3月上旬から発売する．Excel 2.1は，Windows 3.0の特徴である，新しいメモリ管理システムにより，より大きなデータ処理が可能となり，処理速度も大幅に向上した．

価格は，98,000円で，登録ユーザーには無償でアップデートを行う．

ワンラインNEWS
東京電気は東芝のラップトップパソコン「J-3100GT」を親機に使ったPOSシステムを開発・発売した．専用機に比べ価格が安いのが特徴．

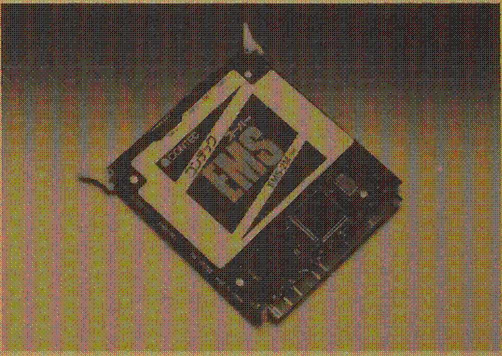
NEWS & NEW PRODUCTS

クレオ，グラフィック，通信機能を備えた低価格ワープロソフトを発売

㈱クレオ（TEL 03-3456-4611）は，同社の株式店頭公開を機に，一般消費者市場を意識したソフトの開発，購入しやすい価格での販売を行うとの営業方針を固め，その第一弾として，グラフィック，通信機能を備えた日本語ワープロソフト「ユーカラWAVE」を9,800円という低価格で発売した．「ユーカラWAVE」は，自動仮名漢字変換を標準装備し，高速表示，アンドゥ，マクロ機能を備えている．カラー36色，最大640×400ドットの絵が作成できるグラフィック機能，ヘイズAT準拠モデムに対応した通信機能ももつ．

コンテック，EMS対応増設メモリボードを発売

㈱コンテック（営業部販売推進課TEL 03-3769-1061）は，NEC PC-9800シリーズ対応のEMS対応増設メモリボード「スーパーEMS」を発売した．スーパーEMSには，専用の設定プログラム「おまかせ設定プログラム」，操作を解説した「おまかせ設定シート」が付属しており，ボード上のスイッチ類も出荷時に設定されている．また，EMS機能以外にもXMS，仮想EMS，仮想UMB，増設メモリなどにも使うことができる．2Mから8Mで，価格は37,000円から115,000円まで．



アメリカン・テクノロジー・グループ，超高速モデムを普及価格で発売

アメリカン・テクノロジー・グループ㈱（TEL 03-3503-4516）は，通信規格CCITT V.32(9600bps)，MNPクラス5により19200bpsのスループットを実現した「MultiModem V32S」を発売した．これは，エラー訂正機能としてCCITT V.42，コーリングシーケンスとしてV.25bis，セキュリティ・コールバック機能，リモート・コンフィギュレーション機能を搭載している．価格は，198,000円．

アイ・オー・データ機器，98拡張用スロットを応用した新規格を発表

㈱アイ・オー・データ機器は，（TEL 0762-21-4812）は，日本電気のPC-9800シリーズ拡張用スロットバスを応用し，拡張スロットを2倍に利用できる「98セカンドバス」規格を提唱した．これはベースボードとサブボードからなり，2つのボードをPC-9800シリーズの拡張用スロット，1スロットに接続し，異なる目的の2枚のボードを同時に使用することができるという規格である．同社では，この規格対応のメモリボードの出荷をすすめるとともに，ほかの拡張ボードメーカーにも広くアピールしていく．

◆ ◆ 佐々木哲のニューヨーク ◆ ◆

N.Y. システム通信

SYSTEM N Y NEWS

春に発表？MS-DOS 5

現在，MS-DOSのバージョン5が準備されています．昨年夏前から広い範囲でベータテスト中です．出版された情報類から判断すると，DOS 5は次のような特徴をもちます．

- DOSフリーエリアの増大．DOS本体をHMA（High Memory Area）に置く．
- ファイルシステムはDOS 4とあまり変わらない．
- XMS（HIMEM.SYS）をサポート．
- UMB（Upper Memory Block）のサポート．DOSのメモリ管理に含まれる．また，LOADHIGHコマンドをサポート．
- ユーティリティ類の強化．
  新しいDOS Shell／フルスクリーンエディタをサポート／QuickBASICを含む／その他ユーティリティ類の追加
- 簡単なタスクスイッチャーをサポート．

幸いなことに，DOS 4のときと違って互換性が保たれているようです．これだけベータテストが進められていればバグも少ないでしょう．IBM主導のDOS 4のときに比べて，Microsoftのほうがまともな仕事をするようで，どのレポートも好意的です．

一般のアプリケーションは安心してそのまま使えそうですね．もちろんあまり変なことをやっていなければという条件つきです．一般への発表は今年の春ころになりそう．まだまだ続くDOSの時代です．

DOSの知られざる世界

世の中からみると非常に簡単なOSであるPC/MS-DOSですが，まだまだ謎がいろいろあります．DOSのユーティリティだけを見てもCOMMAND.COM，SUBST.EXE，JOIN.EXEはいったいどのようにして動作しているのでしょう．マニュアルに書かれていないファンクションがたくさん見かけられます．米国では，まずPRINT.COM（実はPRINT.EXE）の不思議な動きを見て解析を進め，Sidekickに代表されるメモリ常駐ソフトが早くから発展しました．またDOSのメモリ管理を一般に明らかにしたMAPMEM，MARK，RELEASEなどを含むTurboPower SoftwareのTSRCOMユーティリティがあります．

いっぽう，日本ではFEPの絡みもあり，デバイスドライバが非常に発展しました．キャラクタ型デバイスドライバをコマンドラインからロード，アンロードできるユーティリティADDDRV/DELDRV（あるいはPDSのADDDEV/DELDEV）は日本特有です．

これらは，公式のDOSマニュアルに記載されていない部分の解析が進んだ結果です．もちろん，非公開の機能を使うには細心の注意が必要です．米国においてもいろいろ試行錯誤がありました．行儀の悪いアプリケーションは市場から淘汰されていきます．

DOSの情報はさまざまな形で出版されてきました．Ray Duncanの『Advanced DOS』は古典です．Microsoftからも『MS-DOS Encyclopedia』として上下2巻の解説があります．OS/2も噂され，DOSの時代も終わりかといわれていたころで，MicrosoftのDOSの集大成として期待されました．結果として新しいこともなく失望した人も多かったのです．

いま，いちばん完璧なDOSの情報はPDSとしてオンラインで入手できます．Ralf Brownがまとめている Interrupt Listというものです．IBM PC上のありとあらゆる割り込みに関するデータがまとめられています．もちろんDOSに関しても，これでもかというほど集められています．一般にこのファイルはINTER590.ZIPなどの名前で入手できます．数字の部分はリビジョンナンバーです．中身はテキストファイルです．最新ものでは解凍すると1MB近くの大きさになります．そこは情報の宝庫！

非公開情報の決定版

このリストだけでは，使用方法が見当もつかないことがあります．また非公開のファンクションがどの程度信頼できるかが判断しにくいのが難です．オンラインコミュニティではBIXのms.dos/secret会議が非常に多くのことを教えてくれました．

昨年末DOSの非公開の面をまとめた決定版が出版さ

ワンラインNEWS
米半導体検査装置メーカーのKLAインスツルメンツは2年後をメドに日本で生産を開始する．同社は多摩，川崎地区を中心に東京周辺で工場用地を探している．

れました．Addison-Wesleyからの『UNDOCUMENTED DOS』*1です．著者は Andrew Schulman（Pharlap），Raymond J. Michels, Jim Kyle, Tim Patterson（MSDOS 1.x の作者），David Maxey, Ralf Brown（Interrupt List!!）です．みな活発に活躍し実績のある人たちですね．

700ページ近い大冊です．1.2MBのFD 2 枚に膨大なサンプルと上のInterrupt Listのハイパーウェア化されたものが付いてきます．簡単に目次からピックアップすると，

1 章：DOSの非公開機能について（概論に混じって，DOSの全機能のうち 4 分の 1 は非公開だという観察があります）
2 章：公開機能と非公開機能を使ってのプログラミング比較
3 章：MS-DOSのリソースマネージメント，メモリ，プロセス，デバイス
4 章：DOSのファイルシステムとネットワークリダイレクタ
5 章：メモリ常駐ソフト
6 章：コマンドインタープリタ
7 章：MS-DOSのデバッグインターフェース（DOSとWindowsのデバッガのサンプルまであります）
8 章：DOS探検のためのツール，INTRSPY

そして巻末に付録としてInterrupt Listの非公開DOS機能の部分がまとめられています．ここだけで280ページもある！

本文中ではNetware, Windows, DOS Extenderなど実在のシステムとの絡みを含めて，深い議論が，豊富なプログラミングサンプルとともに広げられています．おすすめの本ですね．

この本を読んで感じることは，常識の重要さです．また市場を知り，トータルなシステムの中で使えるソフトを書く重要性です．IBM, Microsoftでさえもこのルールに反すると結果は惨めなものになります．たとえばOS/2のDOS互換ボックスでは，最初はこれらの非公開DOS機能はほとんどサポートされていませんでした．ところが1.1，1.2と進むにつれ追加されていったというおもしろい観察があります．そしてOS/2 2.0では，ほぼ完璧なDOS環境が16個まで動かせるまでになったのです．

*1: UNDOCUMENTED DOS by Andrew Schulman
Addison-Wesley ISBN 0-201-57064-5 $39.95

すばらしいPC上のデバッガ

IBM PC互換機では，デバッガひとつをとっても実に便利なものがたくさんあります．

たとえば，デバイスドライバをどのようにデバッグしますか？　メモリ常駐ソフトでは？　さらには通信がらみのリアルタイムのバグがでたらどうします？　Symdebでは手に負えません．CodeView，PWBは重いし用途が違います．

日本でもBorlandのTurbo Debuggerが紹介されたことでかなり楽になりましたが，まだ不十分です．高価なICEを使いますか？　マニュアルを読んでいるだけでレンタル期間が終わってしまうでしょう．

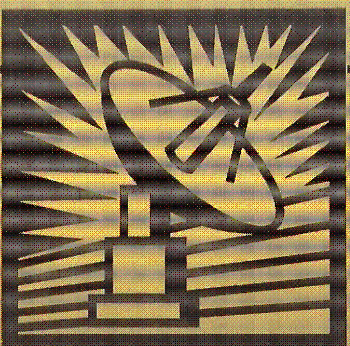
Periscope*2というデバッガがあります．現在バージョン 5 です．DOS上で動作しますが，高級言語のソースレベルデバッギングから，デバイスドライバ，TSR，Plink，.RTLinkのオーバーレイ，Windows 3.0がデバッグできます．DOSそのもの，BIOSだろうがトレースできます．いくつかのモデルがあります．ソフトウェアだけのmodel II-Xからはじまり，NMIを利用した外部スイッチでプログラムを止められるmodel II，デバッガ自体を特殊なメモリボードに移し，システムのメモリを使わないmodel I，リアルタイムでトレースできるmodel IVなどなど．model II，model II-X以外はPCバス（あるいはマイクロチャネルバス）用のボードが付いてきます．

この会社のすばらしいところは，どのモデルを購入してもあとで差額を払えばアップグレードできることです．バージョンアップサポートも親切です．まずは，スイッチ付きのmodel IIあたりを試してみてください．225ドルです．これをいったん使いなれるともうIBM互換機以外では開発したくなくなること請け合い．

*2: Periscope model II－X $195.00,model II $225.00,model I with 512k RAM $595 ($695 for PS/2),model IV $1895-$2395
The Periscope Company, Inc 1197 Peachtree St., Atlanta, GA 30361
phone: (404) 875-8080 fax: (404) 872-1973

「今月の何だこりゃ？」，スペースがないので簡潔に．お休みです．ごめんなさい．

Illustration : NOBUKI FUKUHARA



西川和久の

希望的観測

DOS環境に切り札登場か!?
オープン・アーキテクチャー・デベロッパーズ・グループ設立へ

1 事態は意外な早さで展開!

創刊号で「VGA漢字DOS－IBM DOS J4.0/V（以下，DOS/V)」について述べた．読んだ方には，このDOS/Vがコンピュータ環境の国際化にとって大きな意味をもつことをおわかりいただけたと思う．それがユーザーにメリットをもたらすことも……．

また，「個人的な希望」として，DOS/VをベースにAXやJ-3100などのPC互換陣営が手を結ぶことになれば，国際性をもったマシンが一本化されるのではないかとも書いておいた．この「個人的な希望」は，思っていたよりもかなり早く具体的な動きになりそうである．

本誌の創刊号が出て間もないタイミングで，昨年12月20日に，『日本でのパソコン用アプリケーションの利用範囲拡大を目的に日本アイ・ビー・エム（株）が「オープン・アーキテクチャー・デベロッパーズ・グループ（仮称)」を設立』することを発表した．

このグループは，参加パソコンメーカー各社の間で，OSやディスプレイ制御論理モジュールなどの共通基盤を築き，アプリケーションの異機種間利用を実現しようというものだ．そのために，日本アイ・ビー・エムは，英語と日本語に共通したIBMパーソナルシステムに関する製品情報を公開・提供する，としている．その発表を受けて，同日AX協議会は「オープンアーキテクチャの理念でソフトウェア・ハードウェアを開示するものであり歓迎する」というコメントを発表した．

2 新聞は一番肝心なことを書かなかった

私も発表会場に居合わせたが，そのとき配られた資料の中にとてつもない一節を見つけてしまった．当然，翌日の新聞ではこの件に関していろいろ載っていたが，どれも目先のことばかりで，一番肝心のことにまったく触れられていない．たとえば「AX協議会の加盟各社，東芝とでシェア30%前後のグループが誕生か!?」「プライドを捨てて互換機連合作り」「パソコンで包囲網」など，いかにも日本アイ・ビー・エムと日本マイクロソフトが勝手に国内で決めたかの如く扱われていた……．

ではこの「一番肝心な」そして「とてつもない」一節とはなんであろうか!?　発表時の資料の「具体的な内容」とされている4点をそのまま写すと，

(1) IBMのパーソナル・システムのアーキテクチャーに準拠した製品を開発するメーカー各社に「オープン・アーキテクチャー・デベロッパーズ・グループ」のメンバーとして参加していただき，IBMのパーソナル・システム（PC-ATアーキテクチャー）に関する製品情報およびテスト条件・環境を提供、支援します

ワンラインNEWS
次々世代のメモリである64MbitDRAM向け基本技術の開発が相次いでいる．三菱電機と沖電気がそれぞれメモリセルの小型化に成功した．

(2) IBMが開発し、提供しているパソコン用OSのひとつである「DOSJ4.0/V」を、IBMコーポレーションと米国Microsoft社（ビルゲイツ会長）との基本合意に基づき、日本のマイクロソフト社からOEM供給します

(3) ディスプレイ上に情報を表示するためのIBMの規格であるVGA（Video Graphic Array）に準拠した画面制御用論理モジュールおよび技術情報（テクニカル・リファレンス・ガイド）を日本アイ・ビー・エムから参加各社に提供します．なおこのモジュールは、技術的なサポートも含め、株式会社アスキー（西和彦社長）からも提供される予定です

(4) 今後、OS/2、マイクロチャネル・アーキテクチャー、画面制御理論の拡張等についても順次支援する予定です
（発表当日のプレス資料から抜粋）

この中で，(2)の「IBMコーポレーションと米国Microsoft社（ビルゲイツ会長）との基本合意に基づき……」という一節が「一番肝心」で「とてつもない」のである．これは即ち，『USで本家本元の２社がDOS/Vを認めた』ことにほかならない．創刊号で少し触れた「PC-DOS/MS-DOS DBCS* Version」としてDOS/Vが米国本土で登場する可能性が濃厚になったといえよう．
＊DBCS：Double-Byte Character Set

3 ベルリンの壁は落ちたが……

これは大事件である!!今までDOSの新しいバージョンが出るたびに日本語版は半年以上待たされ（国内では出ないケースもある）じれったい思いをしてきたが，タイムラグ無しに使える可能性が出てきた．また国内の売れているアプリケーション・ベストテンなどを眺めていると，その半分近くが米国版からの移植物である．今後，これらのアプリケーションがはじめからDBCSを考慮し，DOS/Vにも対応した形で市場に出てくる可能性もある．（力のあるソフトハウスはカントリーコードを見て，メニューやヘルプを日本語化し罫線にまで対応してくるかも知れない……）

この推理が正しければ（少し時間がかかるかも知れないが），これまでの国内パソコン業界事情は大幅に変わって来ると思われる．なにしろ主流OSのほとんどすべては米国からきているものであり，その上ではじめから漢字（DBCS）が扱えるとなればこれは画期的なことである．当然のことながらUS-PCとアーキテクチャーが全く違うパソコンは排除されていく方向へ向かうであろう．これで日本もやっと『国際パソコン文化』が開けるわけだ！ただ，発表時に使われた言葉として「ベルリンの壁が落ちた」とあったが，ドイツの場合と同様に後処理のほうが前途多難な様子である……．

どちらにせよ，US-PCフリークの私にとっては非常に嬉しい発表であった…… :-)

Programmer's Rest Bench

「月刊プログラマーズ・ページ」の「プログラマ」とは

■大型機の世界では……

「プログラマ」という言葉は，とくに大型汎用機の経験がある人にとっていいイメージを与えないようである．大型汎用機の世界では，「プログラマ」は「システムエンジニア」に指示されてプログラムをコーディングする人という意味合いが強い．「プログラマはシステムエンジニアになるために通過する職種である」ということだろうか．どうも，プログラムを書く人＝プログラマというように簡単にはいかないらしい．

こうした区別は，大型汎用機の世界が限られたハードウェア・ソフトウェア資産を多くの人間で高度に活用することを前提に成り立っているからであろう．CPU時間を分割してさまざまな使い方をする，そのために分業化してプログラム開発に人員を投入する，という図式になる．そのときに，目的を実現するためにどういうプログラムを作ればいいのかを考え，「プログラマ」にコーディングを指示する人を「システムエンジニア」といい，実際にコーディングする人を「プログラマ」という．

■マイクロプロセッサから始まる世界

パーソナルコンピュータの世界では，「プログラマはプログラムを書く人」という単純な言い方があまり違和感なく受け入れられている．これは，パソコンが文字通り個人のものであり，プログラムも基本的に個人の単位で作るからだろう．

パーソナルコンピュータは，マイクロプロセッサの登場によって成立した．小メーカーでもコンピュータを作ることができるようになったし，アマチュアのホビイストが集まって新機種を組み上げることだってできるようになった．事実，アップル社がスティーブン・ジョブズのガレージで最初のコンピュータを組み立てたことは有名な話である．

パーソナルコンピュータが広く普及して，かつてのようなベンチャービジネスが成立しにくい状況になったが，「パーソナル」という意味は厳然として存在している．

この世界では，プログラムを書く動機，目的，方法などのどれかを，またはすべてをプログラマが把握し，決定することができる．そこに，創造的な新しいプログラマ像を見ることができると考えるのだが…….

SE-NETへアクセスされた読者へ

本誌創刊と同時に運用を開始したソフトウェアオンラインサービスのためのBBS「SE-NET」には，連日たくさんの読者からアクセスをいただいております．「予算の関係」で1回線だけのホストですので，接続できないことも多かったと思います．多くの読者のみなさまにアクセスのお礼を申し上げるとともに，つながらなかった方にお詫びいたします．また，ご不満な点もあるかと思いますが，引き続き使いやすいBBSをめざして整備していきますので，よろしくお願いいたします．シスオペ担当者は，つぎのようにコメントしております．

――SE-NETは現在，本誌に掲載されたソースプログラムの配布を中心に稼働しています．これからはパソコン通信のメリットである双方向性を最大限に生かして，本誌との連携したコミュニケーションを展開したいと考えています．「本誌と読者」や「読者と読者」の楽しい関係を作るために，みなさまの「声」を待っています．

ワンラインNEWS
松下電器産業の子会社であるパネシークエント社は32bitCPUを2から6個接続できる同シリーズで低価格な並列処理コンピュータを発売した．

Illustration：Mikio Sasaki

パソコン博物館

井戸禎光

III

ゲームマシンの先駆けとなった電子帳簿

PET2001

コモドール PET2001

今回は，コモドール社のだしたPETとそのICにスポットライトを当ててみよう．コモドール社はその社長であったジャック・トラミエル氏で有名である．PETを売りだした当時のコモドール社はカナダにあり，電卓で世界を制覇していた．またマスセールスの実現でも優れていた．

トラミエル氏の口癖は，「日本人がやってくる，その前に我々自身が日本人にならなくては」というものであった．もっとも，彼の考えていた日本人像は，品質管理などの面を指していたのではなく，量産と価格攻勢のことを考えていたようである．

さて，1977年7月に発表したコモドール社のPETの価格は595ドル，ディスプレイ込みで800ドルであった．当時の市場がいったいどんなものであったかを，『Personal Computing』誌のインタビュー（トランジスタ技術の1978年1月号に掲載）を参考にして考えてみよう．第1は「一般大衆向け計算機市場は存在しうるのか」，第2は「製造メーカーがそうした大衆向けの価格で高信頼度の要求されるシステムを製作できるのか」，第3は「技術的な背景を持たない一般消費者が，果たして種々の，そして不測の諸問題を克服して，そうしたシステムを使いこなせるのか」というような議論があった．

肯定，否定それぞれに具体的な意見が示されているが，13年たった現在考え直してみると，当っているもの，外れたもの，現在もまだ続いているものなどがあって非常に興味深い．なおIBM社の参入に関しても，「可能性がある」といった意見と「量産するような市場で，コモドール社のような企業には対抗できない」といった意見に分かれていた．

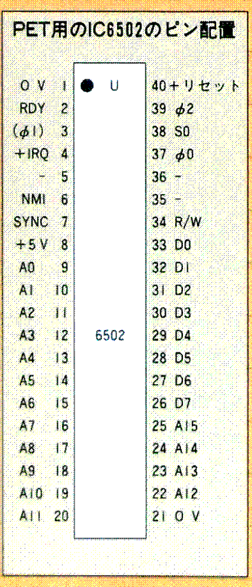
現実には，PET2001はアメリカばかりだけでなくヨーロッパでもよく売れ，とくにドイツでは工業用として，またイギリスではホビイスト用に売れた．PETとは個人用電子帳簿（Personal Electronic Transactor）の略で，OAのはしりである．本体は商店にあるレジスタのような格好で，ディスプレイも独特の

末広がりのような形状だったため，新しいコンピュータを感じさせた．利用にするには８Ｋベーシックとカセットレコーダを使うことになる．スクリーンエディタと逆アセンブラが付いていたし，日本向けにはカタカナROMを売っていたのはよいが，基本的にテキスト中心の設計であった．グラフィックにも重点をおいていたアップルが長寿命だったのに比べて，早く次機種に移っていった．日本で売っていた機種は，ROMが12KBで，RAMが4KBとか8KBのものであった．

PETの系列には，1981年にでたVIC-20という安かったことで有名なもの（当時で300ドル，その後100ドル以下になった）があり，販売量は100万台に達した．1982年にでたコモドール64は，グラフィックスと音を非常によくして500ドルという，安価なマシンであり，PETやVIC-20よりパワフルであった．これらは当初の名前の由来とは異なり，主としてゲーム用に使われていった．アップルが企業での使用を指向しつつ，実際は汎用機としてパソコンを使う個人に根強いファンを作っていったのと対照的である．

Illustraion：Toshihito Ito

このような多量に売れる商品作りを可能にした大きな理由は，中心となるICであるモステクノロジー社の6502CPUにある．6502の系列は好運を背負ったCPUといわれており，前回述べたアップルIIとか，最近では任天堂のゲーム機などに使われて大成功しており，きわめて寿命が長い．

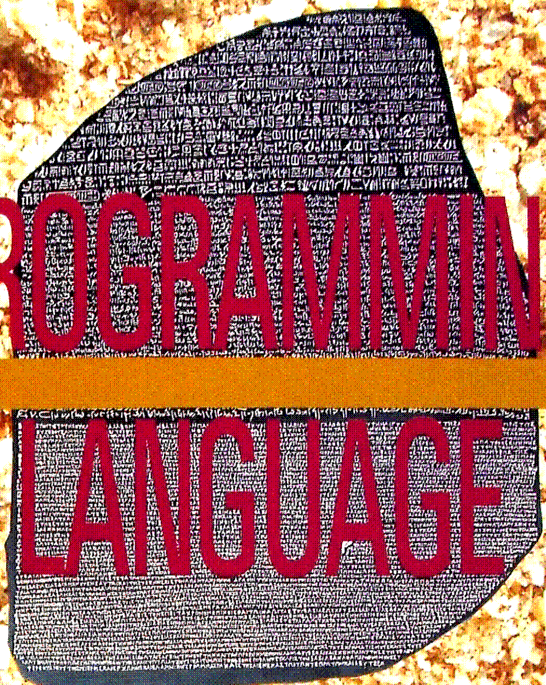


6502を設計したのは，チャック・ペドル氏であり，彼は6502を利用してKIM-1というマシンを作っていた．しかし，彼は自分の会社をコモドール社に売ってしまった．このころのアップル社はまだガレージで仕事をしていた．そしてアップル社には資金がなかった．多くのコンピュータが使っていたCPUは，当時もインテル社の8080であったが，価格が270ドルもしたことと，銀行口座を持った会社でないと売らなかったこともあって，スタートしたばかりの会社では買えなかった．一方，6502のほうは，コンピュータショウの会場で20ドルで買えた．そのためにウォズニアック氏は，6502を使ってアップルを作ることができた．

のちにトラミエル氏はコモドール社を離れて，ノラン・ブシュネル氏の作ったアタリ社を買い，一方，コモドール社のほうはアミガ（現在は68000ベース）を出すことになる．アタリ社は，当時ビデオゲームで成功をおさめていたが，ブシュネル氏は1976年にアタリ社を2800万ドルでワーナーコミュニケーション社に売ってしまっていたので，トラミエル氏はワーナーコミュニケーション社からアタリ社を買うことになった．

結論として，「ゲームは小遣いの範囲で買えなければならない」ということになり，逆にいえば「利用目的がマシンに触ることである機械は，好き者の持つ予算範囲であれば，好き者の人数分だけ売れる」ということになろうか．

PROGRAMMING LANGUAGE



特集・この言語を使いなさい!!

プログラミング言語とはコンピュータに命令を与えるための「言語」である．命令を与えることが目的だから，それが言語である必要はない．事実，ENIACのような，ごく初期のコンピュータではハードウェアの結線を変更することによって命令を実行させていたのだ．しかし，それが「言語」になったのは，効率の問題や人間同士の間でプログラムを理解する必要があるからだ．プログラミング言語が出現して以来，おもに機械を動作させることも含めた「効率」と，人間による「可読性」の向上を求めてさまざまな言語が出現した．だからといって，現在メインで使われている言語がいちばん優れているのか，といったらそう簡単にはいかない．広範に使われる条件には，プログラミング言語の特性のほかにさまざまな環境が加味されるからだ．いったいプログラミング言語は何を求めて作り出され，それを使う人は何を考えているのだろうか．あまたあるプログラミング言語なかから多く使われているもの，特色のあるもの14を選び出し，その言語を使い込んでいる著者に語ってもらった．

パソコン・プログラミング言語盛衰記

パソコンにおけるプログラミング言語の流れは，コンピュータサイエンスにおけるプログラミング言語の発達の歴史とは必ずしも一致していない．これは，初期のパソコン（マイコン）のハードウェアが大型コンピュータに比べて極度に貧弱であったこと，パソコンでプログラミングを行っている人の多くがコンピュータの専門家ではないアマチュアプログラマだったこと，安価に入手できる言語処理系の有無などが原因となって生じたことである．

プログラミング言語の系譜については特集本文でご覧いただくとして，ここではパソコンによるプログラミングを趣味としてきたアマチュアプログラマの目から見た日本におけるパソコンプログラミング言語の移り変わりについて述べてみた．

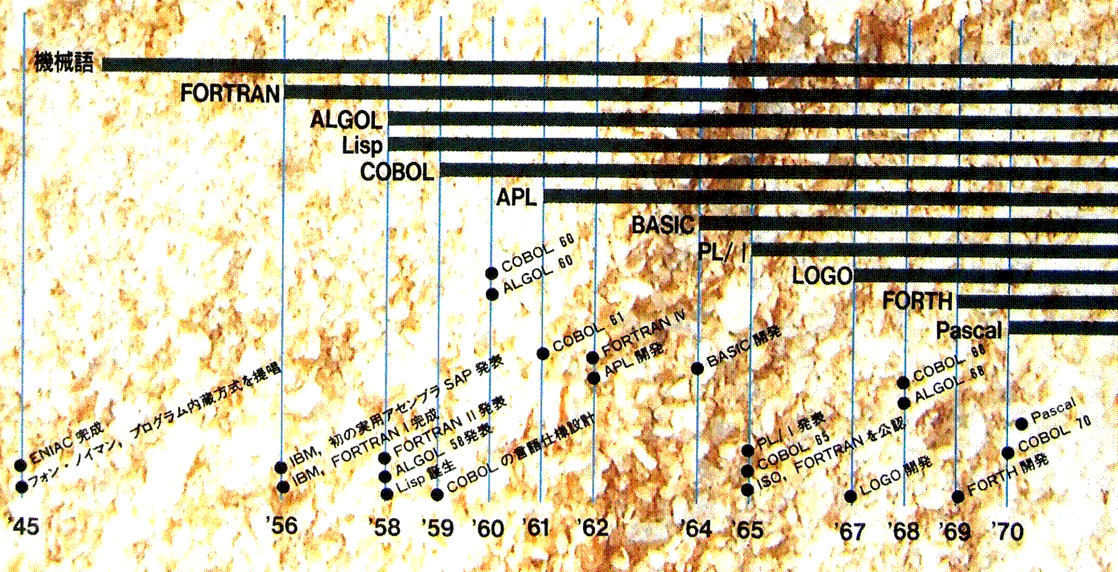
黎明篇

日本におけるパソコンの歴史は，1976年8月に発売された日本電気のTK-80に始まる．このTK-80が予想外に売れたため，翌年には各社から多くの類似製品が発売され，ここにマイコンブームが出現する．

当時はパソコンという言葉はまだなく，これらのコンピュータはその形状からワンボードマイコンまたはたんにマイコンと呼ばれていた．電子部品がハンダ付けされた剝き出しの基板と，0～9の数字キー，A～Fのアルファベットキー，それにいくつかの特殊キーがついているだけのキーボード，8桁程度のLEDディスプレイというのが当時の標準的なワンボードマイコンで，プログラミングはキーボードから直接マシン語コードを打ち込むというものだった．

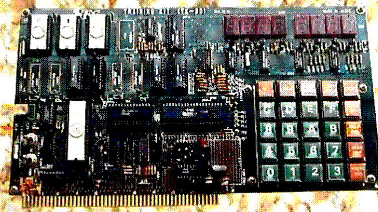
ROM内にアセンブラを内蔵したマシンもあ

プログラミング言語の歴史



るにはあったし，アメリカで発表されたTiny-BASICを改造したワンボードマイコンに移植して使用していたつわ者もいるにはいたが，大半のユーザーはマニュアルに載っているオペレーションを一通り試し終わるとマイコンを放り出してしまった．それというのも，当時は「コンピュータはなんでもできる」という誤った考えが人々の間に広まっており，ユーザーがマイコンに期待するものと現実のマイコンとのギャップが大きすぎたからである．

TK-80（日本電気）



1976年に日本電気から発売されたマイコントレーニングキット．予想をはるかに上回る人気を集め，TK-80を求めるマニアが秋葉原に行列をつくった．ラジオ・無線マニアにとっては新しいオモチャであり，個人で所有できるはじめてのコンピュータでもあった．
■CPU：μPD8080AD（2MHz）■ROM768バイト（1KBまで拡張可能）■RAM512バイト（1KBまで拡張可能）■7セグメント8桁のLED表示装置（16進表示）■25個のキーボードスイッチ（データキー16個，ファンクションキー9個）■CMTインターフェース接続により外部記憶装置としてカセットレコーダー使用可能■標準価格：88,500円

しかしながら，マイコンとマシン語の世界に完全にハマッてしまった人間も少なからず存在し，10数年を経た今，彼らはりっぱな業界人として活躍している．

BASIC勃興篇

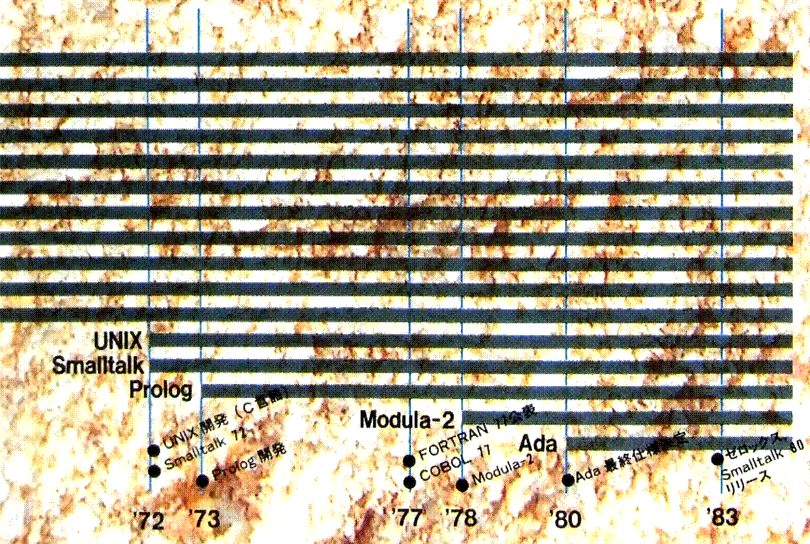
1979年に日本電気からPC-8001が発売されると，パソコンプログラミングの世界はいっきにBASIC一色に塗りつぶされることになる．

PC-8001は，カラーグラフィックが使用できる，豊富な周辺機器がそろっている，マイクロソフト系のBASICをROMに内蔵しているなど，マニアの憧れの的だったApple IIやPETなどと同程度の機能をもちながら，これらのマシンよりずっと安価だったため，たちまちベストセラーになった．

PC-8001以降に発売された各社のパソコンの大部分が，PC-8001と同様にROMにBASICを内蔵していたため，ごく自然にパソコンプログラム＝BASICという図式が完成したのである．

新型マシンが次々に発表され，ついに16ビットマシンであるPC-9801が発売されるに到っても BASICの重要性は変わらず，パソコン雑誌にはBASICのソースリストが何ページにもわたって掲載され，ユーザーはそれを自分で打ち込んで楽しんでいたものである．

このころおきた大きな変化にプログラムの販売がある．ワンボードマイコンでのプログラミングは，あくまでも個人の趣味にしかならなかったが，BASICマシンでのプログラミングは商売になったのである．

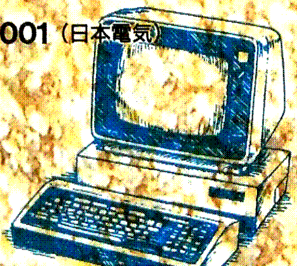


1982年になると，企業のOA化

がブームになり，パソコンとはなんの関係もない一般誌が「これからのビジネスマンは，BASIC くらいできないと生き残れない」などと，とんでもないことを書きたてた結果，会社帰りのサラリーマンがせっせと BASIC 教室に通うという珍現象もおきている．テレビでは，NHK の「パソコン入門」，民放の「パソコンサンデー」が放送されている．

1983年には，その名も「The BASIC」なる雑誌も創刊され，BASIC は全盛を極めることになる．

PC-8001（日本電気）



Illustration：Ruriko Masuda

1979年９月に発売されたパーソナルコンピュータ．当時としては強力な機能を持った N-BASIC を搭載し，外国製のパソコンに十分対抗できる機種だった．PC-8001は，それまでハードを組み立てられるマニアに限られていたマイコンブームを大きく広げる役割を担ったのであった．

■CPU：μPD780C-1（4MHz，Z80コンパチブル）■ROM24KB（N-BASIC 標準添付）■RAM16KB（最大32KB まで拡張可能）■160×100ドットのカラーグラフィックス（８色）■オーディオカセットインターフェース（600bps）■標準価格：168,000円

構造化プログラミング台頭篇

全盛を誇る BASIC に水を差したのが，ソフトウェア危機と構造化プログラミングである．

現在のようにソフトウェアの生産性が低い状況が続くと，数年後には社会が必要とするだけのソフトウェアを生産するプログラマの数がたりなくなるというのがソフトウェア危機であり，ソフトウェアの生産性を高めるのに有効な手法として提唱されたのが構造化プログラミングである．

パソコン上で使用されていた BASIC の大部分がこの構造化プログラミングをサポートしていなかったため，BASIC は構造化プログラミングの提唱者から袋叩きの目に合わされることになる．

いわく，「BASIC でプログラミングを覚えたプログラマは，一生まともなプログラムは書けない」．

こうまでいわれては，さすがに気にしないわけにはいかない．このとき，構造化プログラミングを身につけるのに最適な言語として紹介されたのが Pascal である．しかしながら，CP/M や MS-DOS 上で動く Pascal コンパイラは10万

パソコン・プログラミング言語の歴史

●80年代初期までにアップル II 用，CP/M 用に各種処理系移植される
アップル II：Pascal，FORTRAN，Lisp，FORTH など
CP/M：BASIC，Pascal，COBOL，FORTRAN，C，Ada，Lisp，FORTH，APL，ALGOL など

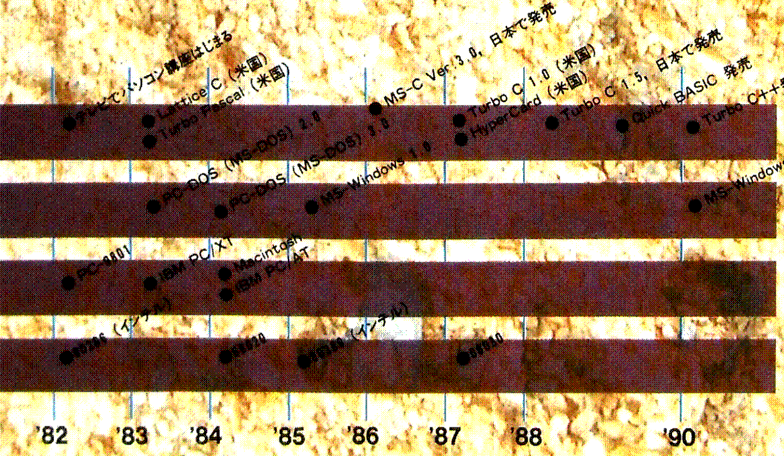
円以上もしたため，そう簡単に購入することはできなかった．

だが，1984年にTurbo Pascalが2万円代で発売されて，この状況は一変する．エディタとコンパイラが結合した統合環境，高速なコンパイル，低価格と3拍子そろったこのTurbo Pascalで構造化プログラミングを身に付けたというプログラマも少なくないはずだ．

Turbo Pascalはちょっとしたブームを引き起こしたが，アメリカのようにBASICをしのぐほどの人気を得るには到らなかった．このすぐあとに起こる巨大なCブームの波にあえなく飲み込まれてしまったからである．

Turbo Pascal Ver. 3.0（ボーランド）

Turbo Pascalは，当時のコンパイラとしては珍しくエディタと連携した統合環境を備えていた．これは，スクリーンエディタ環境で動作するBASICインタプリタになれたユーザーにとって使いやすいものであったに違いない．



C言語篇

C言語のバイブルと呼ばれるカーニハン＆リッチーの「プログラミング言語C」の日本語版が発売されたのは1981年，初代PC-9801が発売される前年のことである．また，MS-DOSが登場したころから，そのMS-DOSが手本にしたUNIXというすごいOSとそのUNIXを記述したCというすごい言語の話はパソコン雑誌でもちょくちょく取り上げられていた．

この業界はいつも先を追いかけている．MS-DOS普及期には，すでに「次期OS」としてのUNIXや「次世代プログラミング言語」であるCの記事を取り上げていたのである．

MS-DOSが普及しはじめると，まずこのMS-DOS上のアプリケーションを開発するソフトハウスの多くがC言語を採用するようになる．この理由としては，分割コンパイルが可能なこと，アセンブラに近い記述が可能で高速に動作するアプリケーションが開発できることなどが上げられるだろう．さらに，MS-DOSを開発・販売しているマイクロソフト社が自社の開発言語としてC言語を採用したことが決定打となり，以後MS-DOS上のアプリケーション開発にはC言語を用いるのが常識となる．

パソコン雑誌の記事も年を追うごとにC言語関係が多くなり，誰もがC言語でプログラミングをしてみたいと考えていたが，Cのコンパイラは20万円から30万円以上もしていたため，アマチュアには手の出るものではなかった．

パソコンがBASICシステムからMS-DOSシステムへと移行し

「プログラミング言語C」(共立出版)

B.W.カーニハン/D.M.リッチー著, 石田晴久訳
C言語のバイブルいえる定番書籍. 1981年に発行されて以来, 1989年に第2版が出されるまで, 百数十刷を数えた大ロングセラー.

ていった時期は, 一種の混迷期であったと考えることができる. プロはともかく, アマチュアプログラマにとってCは高根の花であったし, ほかにセンセーショナルな動きを見せている言語もなかった. なによりも, BASICインタプリタに慣れたものにとって, MS-DOS上のコンパイラへ移行するにはかなりの動機づけが必要だったのだ. Turbo Pascalが勢いを得たのはこのころである.

しかし, 1987年にLet's Cが2万5千円という超低価格で発売されてからいっきに状況は変わった. Let's Cに続けとばかり, Lattice C/パーソナル, Datalight-C OPTIMUM, そして低価格Cコンパイラの本命と目されていたTurbo Cが発売されるにいたって, C言語はアマチュアからプロまでパソコンプログラマが使用する標準言語としての地位を確立したのである.

BASIC蘇生篇

構造化プログラミングの提唱以来, なにかというとけなされてきたBASICだが, 1989年に, マイクロソフト社からQuickBASICが発売されたのを機に復活したようである.

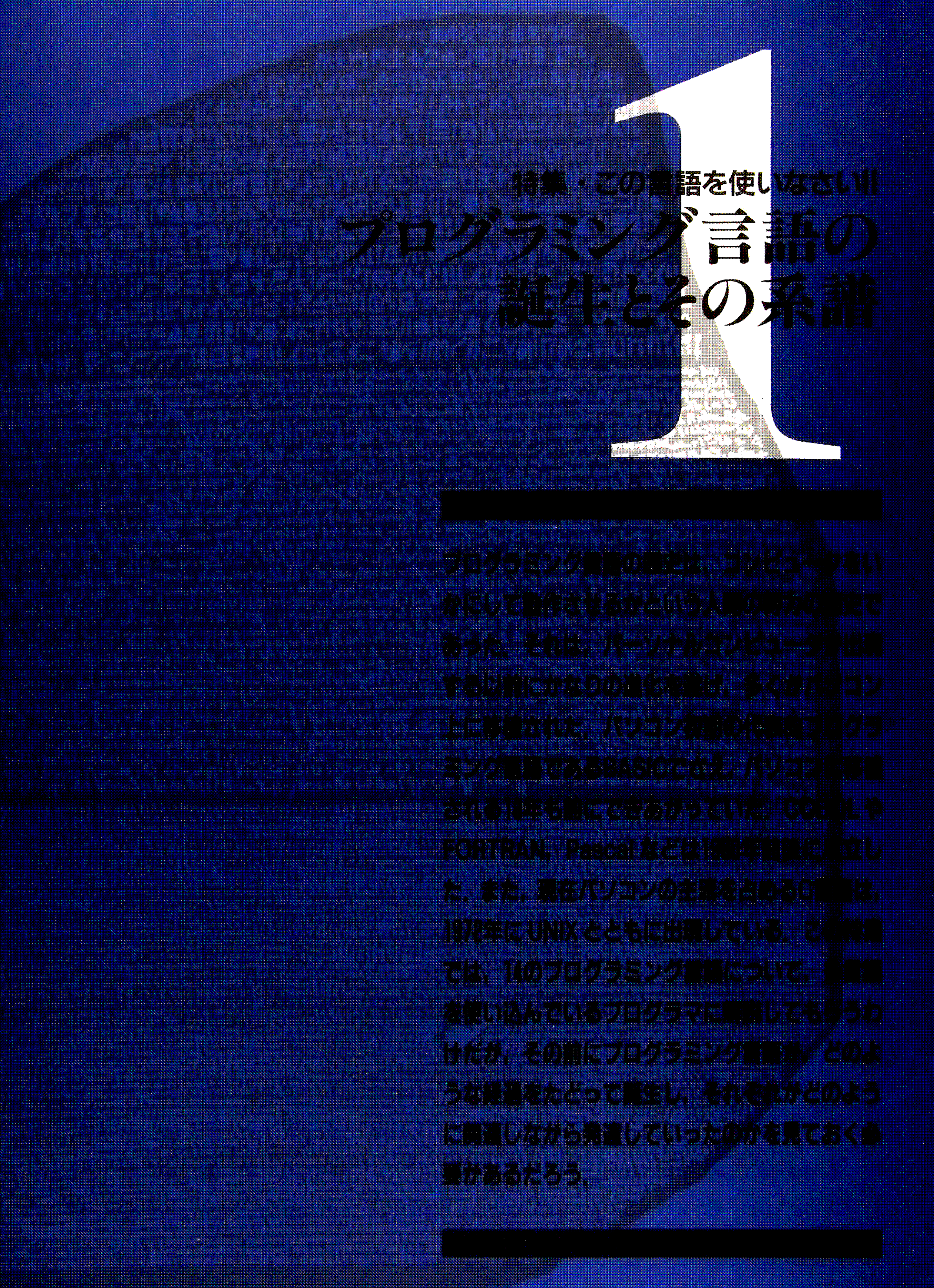
QuickBASICは, 構造化プログラミングのサポート, 行番号を必要としない, インタープリタではなくコンパイラである, Turboシリーズのような統合環境を備えているといった多くの特徴をもっている. これらの特徴から, QuickBASICが, なにかと批判される旧BASICとはまったく別物であることはあきらかである.

QuickBASICによって蘇った新しいBASICは, これからもアマチュアプログラマのよき道具として使われていくことだろう.

QuickBASIC Ver. 4.2 (マイクロソフト)

低価格で使いやすいBASICコンパイラの登場は, かつてのBASICブームでプログラムに触れたユーザーの支持を受けた. 構造化プログラミングのための仕様拡張をおこない, 統合環境とコンパイル・リンク機能を持ちながら, インタプリタ機能も残している. 日本でのリリースはVer. 4.2が最初だった.

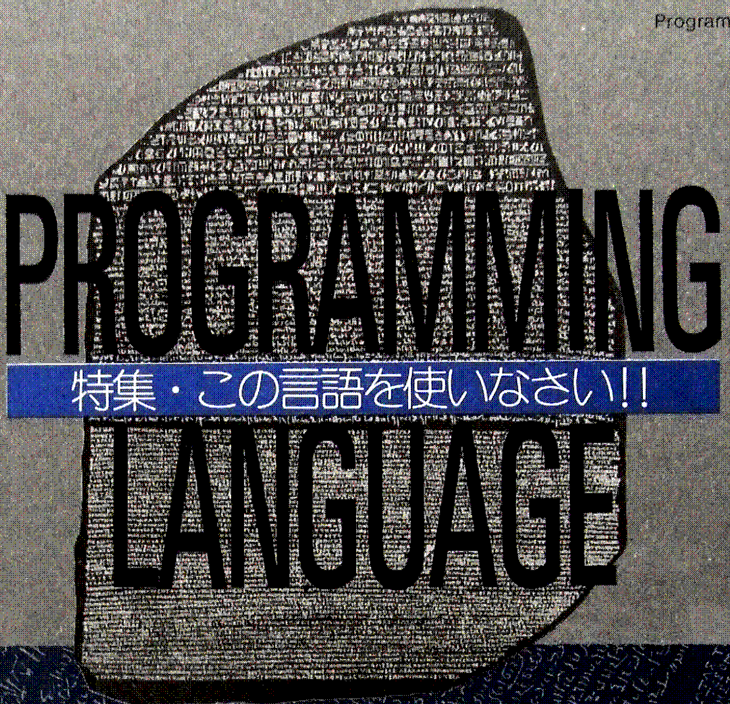
振り返ってみると, パソコンのプログラミング言語の流れは, 大きく3期に区分することができる. マシン語プログラムよりもハードを工作することが主体だったボードマイコン期, パソコン, OAブームの中でのBASIC黄金期, MS-DOSが標準的になってからのC言語期の3つだ. 次にくるのはC++などのオブジェクト指向言語の時代だろうか. いずれにせよ, それによって記述されるプログラムが動作する環境もMS-DOSからWindows, PMなどへと変遷していく.



特集・この言語を使いなさい!!

1 プログラミング言語の誕生とその系譜

プログラミング言語の歴史は、コンピュータをいかにして動作させるかという人類の努力の歴史であった。それは、パーソナルコンピュータが出現する以前にかなりの進化を遂げ、多くがパソコン上に移植された。パソコン初期の代表的プログラミング言語であるBASICでさえ、パソコンが登場される10年も前にできあがっていた。COBOLやFORTRAN、Pascalなどは1960年前後に成立した。また、現在パソコンの主流を占めるC言語は、1972年にUNIXとともに出現している。この特集では、14のプログラミング言語について、その言語を使い込んでいるプログラマに解説してもらうわけだが、その前にプログラミング言語が、どのような経過をたどって誕生し、それぞれがどのように関連しながら発達していったのかを見ておく必要があるだろう。



プログラミング言語の仲間たち

その誕生から系譜を辿る

神林 靖

プログラミング言語の誕生

この世にコンピュータが出現して以来，これに対して作業を与える媒体としてプログラミング言語は発達してきた．米国国防省御用達として名高いプログラミング言語Adaは，世界最初のプログラマであるエイダ・ラブレス伯爵夫人にちなんで名付けられたわけだが，彼女がプログラミングしたコンピュータ，未完に終わったチャールズ・バベッジの解析機関のことを思うとコンピュータとプログラミング言語は鶏と卵の関係にあるともいえよう．

現実に動作するコンピュータの最初のものはペンシルバニア大学のENIACであるが，これにはスイッチやプラグボードによって計算手順が入力されたので，今様のプログラムは存在しない．今日のプログラム内蔵型コンピュータの最初のものはケンブリッジ大学のEDSACといわれているが，そのプログラムを見るとあたかも今日のマイクロプログラムのようである．

当時は，今では考えられないくらいに計算資源は貴重で機械は非力であった．どのように労力を費やしても，効

プログラミング用語
ANSI X 3.159 で規定されるCプログラミング用語を集めました．

率のよいプログラムを書くというのが至上命令であったわけである．それがどれくらいたいへんな作業であったかを物語る例を１つだけ示す．図１はIBM650のプログラムの一部であるが，命令の最後に次に実行する命令のアドレスがあるのに注目されたい．このプログラムは，ドラムという回転する磁気媒体上に置かれて逐次読み込まれながら実行されたのだが，１つの命令の実行が終了したときにドラムの読み込みヘッドの真下に次の命令を置くという最適化プログラミングを行えるようになっている．つまりプログラマは各々の命令の実行時間とドラムの回転時間を測って，次の命令のアドレスを決めなければならなかったわけだ．

図１　IBM 650 プログラムの一部

LOC	OP	DATA	INST	コメント
1107	46	1112	1061	ループボックスを使用するか？
1061	30	0003	1019	
1019	20	1023	1026	アドレス　1023へストア
1026	60	8003	1033	
1033	30	0003	1041	
1041	20	1045	1048	アドレス　1045へストア
1048	60	8003	1105	
1105	30	0003	1063	
1063	44	1067	1076	オペレーション02が要求されているか？
1067	10	1020	8003	
8003	69	8002	1061	サブルーチン01へ分岐

このようなプログラムを当時は２進数または16進数で書いたのだが，すぐに限界に到達することは容易に想像がつく．たとえばわずかな部分を手直ししただけでも全体のアドレスが影響を受けるため，すべてを１から書き直さなければならない．そのような混沌に救世主として登場したのがアセンブラ言語である．これはひとことでいうとアドレスに名前をつけてそれで参照できるようにしたものである．それだけでどれだけプログラミングが楽になったかは，先の例を考えればすぐにわかる．プログラムの一部手直しによって生じるアドレスのずれを気にしなくてよいのである．また変数も意味のないアドレスの数字でなく(多少は)意味のある名前を付けられるようになった．

FORTRANとその子供たち
手続き型言語の系譜

FORTRAN
世界最初の高級言語

FORTRANは，IBMのバッカスらのチームにより18人年をかけて1957年にIBM704上に完成した世界初の高級言語である．当時のハードウェアはまだ非力であったため，FORTRANでも効率的な機械語を生成することが主目的とされている．データ構造は名前にアドレスをわり付けるものであり，基本的にアセンブラと変わらない．コントロール構造は，GOTO指向言語といわれるくらいGOTO文に依存していてIBM704の機械語に適したものとなっている．

図２　アセンブラによるループの記述

```
    I ← 1
LOOP
      ...
    I ← I + 1
    IF I ≦ N THEN GOTO LOOP
```

(a)

```
    I ← 1
LOOP
    IF I > N THEN GOTO OUT
      ...
    I ← I + 1
    GOTO LOOP
OUT
```

(b)

一時構造化プログラミングがブームになったとき，GOTO文が諸悪の根元のようにいわれたことがあった．今日ではGOTO文は必要悪（つまり言外になくてはならないもの）と考えられているが，静的構造と動的構造を混在させていることには違いがない．

FORTRANはIBM704のアーキテクチャを意識した言語ともいえるが，この点現在流行のCも同じようにPDP-11のアーキテクチャを反映したものである．CのほうがFORTRANよりスマートに見えるのは，IBM704とPDP-11の時代の差であろうか．

繰り返しのためのDO文では，ループが必ず１回は実行されなければならないという不思議な構造も機械語への翻訳を考えるとよく理解できる．FORTRANのDO文をアセンブラに書き直したのが図２の（a）である．

プログラミング用語
アーカイブ　バックアップのこと．補助記憶装置に退避する際，容量を確保するために圧縮される．

（b）は0回のループを許す構造にしたものであるが，GOTO文が1つ増えてしまっている．

このように効率的な機械語生成を意図したために，FORTRANはプログラミングしづらい言語になってしまった．空白に意味をもたせないことと変数を宣言しなくてもよいというのも，わかりにくいエラーを作り出す原因になっている．かつて，NASAの金星探査機ビーナスが制御不能に陥ってしまったのも空白にまつわるミスであった．たとえば，次の文では変数DO100Iが生成され，それに実数1.2が代入されてしまう．

DO 100 I ＝ 1 . 20

FORTRANの優れた点は，算術式を通常の記法で書けるようにしたことである．これ以後の言語では，LISP，APL，Smalltalkなど一部の例外を除いて皆これに倣っている．機械語やアセンブラで数値計算を行ったことのある人ならば，これがどれだけの偉大な進歩であったかは容易に想像がつくであろう．

AlgolとPascal

構造化プログラミングを実用化

オペレーティングシステムやアーキテクチャの教科書を読むと，必ずといってよいほど「Algol風の言語で説明する」と書いてある．その割には，Algolの実物を見る機会は少ない．この理由としては，Algolの仕様書がAlgol-58として発表されたときに大勢の人々が感激のあまりすぐに実現に走ってしまい，結果的にばらばらの処理系があちこちでできあがってしまった，IBMのバッカスが中心人物の1人であったにも関わらずIBMが処理系を提供しなかったなどが考えられる．

実用言語としての成功は得られなかったものの，Algolが後世に与えた影響は大きい．世界中の教育機関で使用されているPascalとModula-2がAlgol直系の子孫であるのみならず，今日使われている手続き型言語はすべてAlgolにその源を発している．高級言語の真の歴史は，Algolからはじまったといっても過言ではない．FORTRANはそれに比べるとマクロの完備したアセンブラに近い．

Algolのプログラミング言語の世界への貢献としてブロック構造の発明があるが，それがどのようなものかスタックを用いた手続き呼び出しとスコープルールを通して，Pascalも含めてAlgol系の言語一般について述べてみよう．

図3のPascalプログラムを見てい

図3　ブロック構造とスコープ・ルール

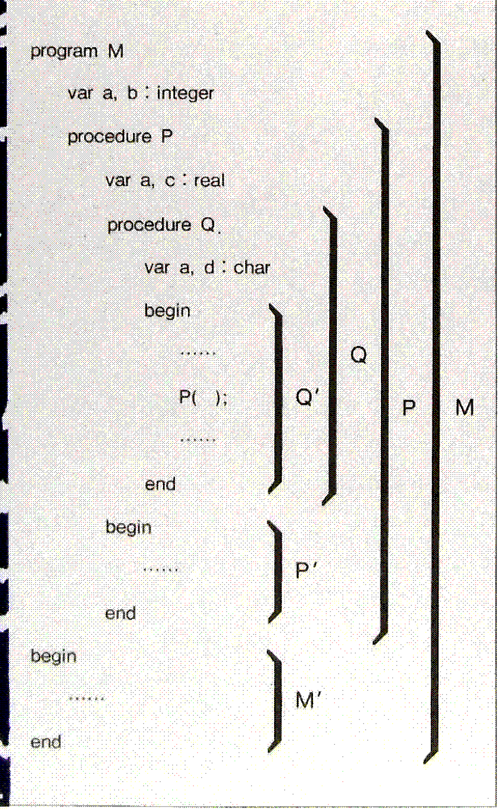
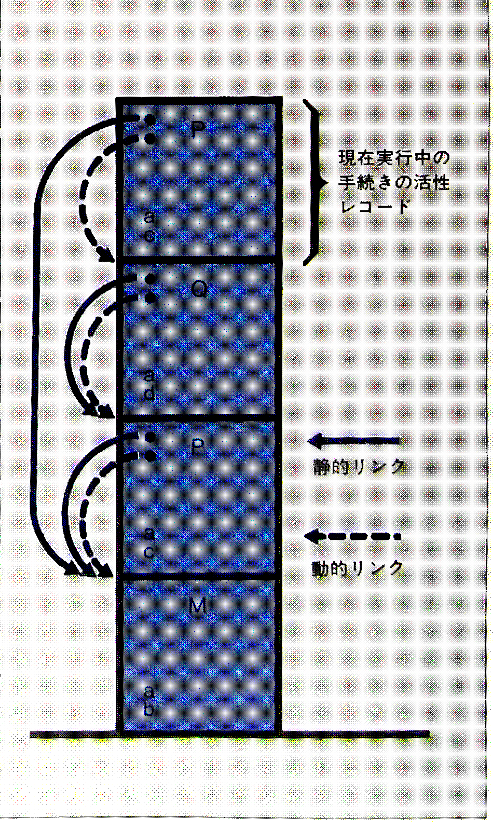


図4　活性レコード



プログラミング用語
ASCII　American Standard Code for Information Interchange（米国標準情報交換用符号）の略．7ビットで表現できる文字セットのこと．

ただきたい．変数 a がプログラムM，手続きP，手続きQの中で宣言されているが，それぞれまったく別のもので，a という名前はM'の中では整数型を指すが，P'の中では実数型，Q'の中では文字型の変数を指す．図中のブロックQ，P，Mをスコープと呼び，その中で宣言された変数はその中でのみ有効で，どのように値が変化しようとも同じ名前をもつ別のスコープの変数には影響を与えない．

ただし，内側のスコープは外側のスコープに含まれていることから想像がつくように，たとえばQ'内で変数 b や c を参照することはできる．このようなプログラムの構造をブロック構造と呼び，これを用いることによりいわゆる構造化プログラミングが可能になった．

ブロック構造を支えているのが，スタックによる手続き呼び出しである．図4は，先のプログラムのQ'内で手続きPが呼び出されたそのときのスタックの状態を示している．各手続きが呼び出されるたびに，活性レコードというものがスタックに積まれる．活性レコードは変数のほかに静的リンク（図中に実線で示す）と動的リンク（図中の破線）をもち，静的リンクを辿ることによりスコープの外側の変数をアクセスすることができる．内側のスコープにある変数をアクセスできないことは明らかであろう．あるブロックが実行されているときには，それより内側の活性レコードは存在しないのだ．

動的リンクは，手続き呼び出しから戻る際に使用される．記憶域を動的に使うスタックを用いた手続き呼び出しによるスコープとブロックの考え方によって，FORTRANでは不可能であった再帰定義なども可能となり，人間が思考するアルゴリズムを自然な形でプログラムに表現できるようになった．

図5　参照呼び出し

```
CALL FIDO(42, A(I))
  ...
FIDO(Y, Z)
  ...
    Y = Y + 1          (Y = 43)
    Y = 5              (Y = 5)
```

図6　名前呼び出し

```
get_x(random(0,1);
    ...
begin
    get_x(x)
    y := x;            (y = 0 または 1)
    z := x + 1;        (z = 1 または 2)
```

手続き呼び出しに関して，FORTRANは参照呼び出しという方法を採用している．これはパラメータのアドレスをサブルーチンに渡すので，パラメータの値をコピーして渡すPascalやCの値呼び出しに比べると効率的である．もっとも，危険もつきまとう．図5のプログラムの断片を見ていただきたい．Yのアドレスと42のアドレスは同じである．したがって，呼ばれたプログラムの中でY＝5が実行されると上書きされてしまう．Pascalで値呼び出しと参照呼び出しの構文を変えたり，Cで値だけを与えるようにし，特別な場合にアドレスを渡せるようにしてあるのは賢明な設計判断であるといえよう．

Algolでは，現在あまり使われなくなった名前呼び出しという手法が使われている．図6のプログラムの断片を見ていただきたい．名前呼び出しではthunkという値評価のためのコードが生成され，渡されたパラメータが参照されるたびに値の評価が行われる．つまり，この例でいうとxは参照の際にランダムに0か1をとるわけである．これは，ちょうどxをrandom(0,1)で置き換えたのと同じ効果があるため，テキスト置換と呼ばれることもある．

Ada
米国国防省のお墨つき

Adaは，1975年から1980年にかけて米国国防省後援のプロジェクトとして開発された．つまり国防省が，Pascal風であること，モジュール構造を支援すること，リアルタイム機構があることなどの基本的な要求仕様をまとめ，応募各社が知恵を絞ったわけである．結局，フランスのCIIハネウェル・ブル社のチームによって作成された言語仕様が競争に勝ち残った．

Adaは，主としてミサイル誘導システムのようなリアルタイムの組み込みシステム用開発言語として意図されたもので，米国では1987年に国防省がAdaの全面的使用を義務づけたのをはじめとして，NASAでも宇宙ステーションのプロジェクトでの使用を義務づけている．民間では，ボーイング社が747-400の飛行制御プログラムにAdaを採用するなど熱心さが目立つ．

プログラミング用語
ANK　Alphabet（英字）・Number（数字）・Kana（カタカナ）を意味し，8ビットで表現できる文字セットのこと．

ヨーロッパ各国の国防省でもAdaの採用を正式決定している。日本では、NTTが商用化しているほかJISの翻訳規格も制定されようとしている。

Adaは大規模なプログラム開発用言語で、これまでに開発された手続き型言語のすべての機能を含んでいる。Algolからの伝統であるプログラムとデータの構造化を助ける多くの機能をもっているだけでなく、Pascal以来、多く取り入れられるようになった強いデータ型やデータ抽象機構によって、データ構造の論理的属性を実現の詳細から分離することができる。また汎用体型の概念によって、処理されるデータの型から独立してプログラムモジュールを生成することができるようになっている。

さらに、モジュールの分割コンパイルとモジュールライブラリ管理機構、そして並列処理のためのタスク機構も備えている。加えてAdaは最初からプログラム開発環境を言語仕様と同時に開発し、言語のみならず支援環境をも統一しようとする点が注目されている。

図7は、3つの入力信号を収集し、それらを用いてある処理を実行する手続きをAdaで記述したものである。それぞれの入力信号は並列に処理される。entry...accept構造で示されるのがランデブーと呼ばれるもので、これにより各タスクは直接通信しあうことができる。タスクが手続き本体中のランデブーで呼び出される前に信号をもつと、呼び出しが到達するまでランデブーで待機状態になる。同様にして信号が整う前に呼び出しがランデブーに到達すると、呼び出すプロセスは、呼び出したタスクが信号を発信するまで待つ。3つの信号すべてが到着すると、手続きprocess_signalが呼び出される。

図7　信号処理のAdaプログラム

```
procedure signal_processing is
    task get_signal_1 is
        entry receive(signal_1: out signal);
    end get_signal_1;
    task body get_signal_1 is
        receive_signal: signal;
    begin
        received_signal := signal_channel_1;
        accept receive(signal_1: out signal) do
            signal_1 := received_signal;
    end get_signal_1;

            ...

begin
    get_signal_1.receive(sig_1);
    get_signal_2.receive(sig_2);
    get_signal_3.receive(sig_3);
    process_signal(sig_1, sig_2, sig_3);
end signal_processing;
```

Adaは、大きすぎること、構造が風変りであることなどの点で多くの人々に批判されている。また、プログラムモジュールの正当性を証明する形式的手法に適応するように設計されてはいるものの、これを大規模に実行する前に行わなければならない準備作業も容易でない。後述する関数型や論理型あるいはオブジェクト指向言語の支持者からは、すでに時代遅れの言語であるとの批判もあるが、手続き型言語の集大成として、またプログラム開発環境を仕様の一部としてもつことなどによりソフトウェア工学からAdaへの期待は大きい。

Occam
並行処理プログラミング言語

Adaがひたすら各種機能や機構を取り込んだ言語であるのに対し、その名前が示す通りOccamは簡潔性を追求した言語である（ウィリアム・オッカムは、唯名論を主張しスコラ哲学の正統を批判して中世から近代への思想的転回点となったオックスフォード学派の中心的思想家。「仮説は必要以上に増やしてはいけない」という提言は、オッカムの剃刀として有名）。

オックスフォード大学のトニー・ホーアによって開発されたOccamは、Adaがめざすのと同じく、多くのプロセッシングエレメントが独立して動作しつつ、相互に協調するコンピュータシステム上での並行処理を支援すべく設計されたプログラミング言語である。Ada同様リアルタイムの組み込みシステムに使うことができるが、Occamの剃刀を用いてシステムから過剰な詳細さをすべて切り捨ててしまっているので、はるかに簡潔な言語となっている。

Occamは、コミュニケーティング・シーケンシャル・プロセス（CSP）のアイデアに基づいている。プロセスは一連の動作を実行し、かつ定義されたチャネルを通じてほかのプロセスと通信を行う。チャネルを通じての通信は同期方式である。つまり、入力と出力

プログラミング用語
ANSI　American National Standards Institute（米国規格協会）の略称。発行した規格は9000件以上。

のプロセスがともに同一チャネルで通信準備完了になるとデータ項目が伝送され，その後は各プロセスとも独立して処理を続行する．

LISPとその子供たち

関数型言語の系譜

関数型言語は，チャーチのラムダ計算にその起源をもつプログラミング言語であり，いかにして結果が計算されるかを指定するアルゴリズムに注目する従来の命令型言語と好対象をなしている．関数型言語は，処理の詳細なステップにあまり深くとらわれることなく計算結果そのものに焦点を当てる．関数型言語のプログラムは，データ値を得るための「処方」の集合ではなくデータ対象の記述の集合とみなされる．関数型プログラミング言語によるプログラムは完全に関数だけから構成され，手続きはまったくない．

代入は非破壊的，つまり変数はいったん値を得たならばそのモジュールを通じて値をもち続ける．これはいかなるふたつの代入文も左辺に同一の変数名をもたないことを意味する．したがって，関数型言語の文は逐次的に解釈される必要はない．すべての文が同時に成立するのである．このため，関数型言語にループ構造はなく，すべての繰り返しは再帰定義を用いる．非破壊的代入と文の非逐次的実行の性質により，新しいアーキテクチャに適した言語となっているだけでなく，関数型の構造という制約によって自然とプログラムは正当性の証明に適したものとなっている．これから実用化への発展が楽しみなプログラミング言語群である．

LISP
人工知能向け非手続き型言語

リスト処理言語LISPは，ジョン・マッカーシーに率いられたMITの人工知能研究グループにより1959年に開発され，記号操作や再帰を含む問題に対して広く使用されてきている．ラムダ計算のアイデアから生まれた言語ではあるが，開発直後からユーザー（FORTRANプログラマ）の強い要望で手続き型言語の機能が取り入れられてしまったため，現存する処理系は不本意にも関数型とはいえなくなってしまった．近年注目を浴びているSchemeというLISPの亜種は，純LISPへの回帰とみることもできる．

LISPの基本データ要素はアトムで，アトムは括弧によってリストとなる．たとえば(A B C)は3個のアトムよりなるリストであり，(A (B C) D)は，2個のアトムと1つの部分リスト（B C）を含むリストである．このようなデータ構造はS式と呼ばれる．

LISPの処理は関数によって記述される．図8に5つの基本的な関数を示す．carおよびcdrという関数名は，LISPが実装された最初のコンピュータのアセンブリ言語の命令に由来している．すなわち，LISPが実装された最初のコンピュータはIBM704であり，命令語はアドレスの低いほうからprefix, decrement, tag, addressと4つのフィールドに分かれていた．そして，carは「Contents of the Address part of Register number」，cdrは「Contents of the Decrement part of Register number」の略称であったといわれている．便宜的に使われた名称が後世に大きな影響を及ぼしてしまった好例といってよいであろう．なお蛇足だが，prefixとtagにもcpr, ctrという関数が定義されていたそうである．

LISPのプログラムでは関数自身S式として記述されているため，ほかの関数によって処理が可能である．それゆえにLISPプログラムは際限ない自己変更を行うことができる．

S式で書かれた関数の例を図9に示す．この関数は，atom_1がlist_2に所属しているか否かを判定している．list_2が非空で，そしてatom_1が

図8　LISPの基本関数

```
(car '(A B C)) = A                     リストの頭部の取り出し
(cdr '(A B C)) = (B C)                 リストの尾部の取り出し
(cons (car '(A B C)) (cdr '(A B C))) = (A B C)
                                       リストの結合
(eq A A) = T                           （真）アトムの同値性の検査
(atom '(A B C)) = Nil                  （偽）アトムか否かの検査
```

図9　アトムの所属を調べるLISPプログラム

```
(defun member (atom_1 list_2)
    (cond ((null list_2) Nil)
          ((eq atom_1 (car list_2)) list_2)
          (T (member atom_1 (cdr list_2)))))
```

プログラミング用語
インクルード　#include前処理指令などでヘッダやソースファイルをプログラムに挿入すること．

list_2の頭部と等しければ関数は真となる。さもなければ，atom_1がlist_2の尾部に所属しているか否かを判定するために再び関数が呼び出される。list_2が空となってしまえば関数は偽となり値Nilによって表示される。

LISPは長年にわたって人工知能プログラミング言語の主流であり，衆目を集めるようなプログラムは例外なくLISPで書かれてきた。またそればかりでなく，LISPは多くの近年の非手続き型言語の出発点となってきた。新しいプログラミングパラダイムあるいはスタイルはすべてLISPから始まったといってよいであろう。

Hope
現在も活躍する関数型言語

現在活躍している関数型言語の例として，英国のエディンバラ大学で開発されたHopeを紹介しよう。

Hopeによるプログラムは関数であり，LISP同様その定義を完遂するためにほかの関数を呼び出すようになっている。関数は汎用体データ型として定義され，特定の型はパラメータとして渡すこともできる。これによりたとえば同一の関数で文字列の集合も数の集合も整列することができる。

Hopeのすべての変数はそれらが定義された関数宣言に対して局所的，つまりプログラムはほかのモジュール内にある変数の値を不当に変更してしまう副作用から免れている。

Hopeプログラムは参照の透明性という数学的属性をもつので，逐次的に解釈される必要はない。参照の透明性は，プログラムの正当性についての言明の構成や証明を容易にするだけでなくプログラムの自動変換をも可能にする。

順次リストの中の正しい位置に要素を挿入するHopeの関数を図10に示す。

1行目は，insertと呼ばれる関数が，未定のデータ型alphaの1個のデータ項目および同型のリストへ作用し別のリストを生じることを宣言している。2行目は，項目iと空リストが与えられたときの関数の値を，項目iに空リストが続くものとして定義している。3行目は，項目iおよび先頭がhで尾部がtであるリストが与えられたときの関数の値を2つの場合に分けて定義している。すなわちiがhより小さければ，関数は，頭部がiで尾部がh：tであるようなリストを生じ，さもなければ頭部はhのままでiを尾部tへ挿入するために関数が再帰的に呼ばれる。この関数が呼び出されると，どちらの定義を使うべきかの判定が並行して行われる。同様に，thenとelse構造に対応する選択定義も並行して評価されるので，比較の結果が判明した時点ですぐに求める値を選ぶことができる。

関数は多相型なので，特定されていない型のデータ項目を操作することができる。

図10　要素を挿入するHopeのプログラム

```
dec insert: alpha # list(alpha) -> list(alpha)
--- insert (i, nil)        <= i :: nil;
  --- insert (i, h :: t) <= if i > h
                              then i :: (h :: t)
                              else h :: insert (i, t);
```

プログラミング言語の鬼子
論理型言語PROLOG

Prologは，アラン・コルメラウアとその同僚によってマルセイユ大学で1972年に開発された。この言語は，制御された演繹推論としての計算の一般的アイデアと，論理と制御の組合せによるアルゴリズムに基づいている。わかりやすくいうと，「AならばBである」と「BならばCである」が「AならばCである」を導く日常的な三段論法を基本演算としている言語なのだ。

Prologでは，従来のプログラミングにおける処理とデータ構造の両側面を論理に基づく単一の記法によって表現できる。「Aである」のような表現を述語と呼び，「AならばBである」のような表現をホーン節と呼ぶ。述語中のデータは単一の項目でもよいし，リストでもよい。たとえば，リストの所属は図11の節によって定義される（いい換えれば，データ項目xは，それがリストの頭部であるか，もしくはリストの尾部に所属するとき，リストに所属していることになる）。

図12に，簡単な化学データベースを使用するPrologプログラムの例を示す。最後の節は，yが要素zのリストをもち，かつxがリストzに所属していれば，xはyの構成要素であると解釈される。

Prologの処理系は，ユーザーからの質問を受けてこれらのホーン節から推論を行う。たとえば，「どの化合物が，炭素を構成要素としてもつか」という質問も図13のような述語として記述

プログラミング用語
インデント　行の先頭にタブや空白を入れること。プログラミングスタイルのひとつでソースファイルでプログラムの流れを判断しやすくする。

図11　リストの所属

```
member-of(x, (x | z))
member-of(x, (y | z)) if member-of(x, z)
```

図12　化学データベースを使用する Prolog のプログラム

```
elements-of(water, (hydrogen oxygen))
elements-of(ammonia, (nitrogen hydrogen))
elements-of(methane, (carbon hydrogen))
elements-of(carbon-dioxide, (carbon oxygen))
elements-of(alcohl, (carbon hydrogen oxygen))
member-of(x, (x | z))
member-of(x, (y | z)) if member-of(x, z)
component-of(x, y) if elements-of(y, z) and member-of(x, z)
```

図13　ユーザーの質問を述語として記述

```
which(y, component-of(carbon, y))
```

図14　述語の結果

```
methane
carbon-dioxide
alcohl
no (more) answers
```

図15　無限ループを生じる節

```
linked-to(x, z) if linked-to(x, y) and linked-to(y, z)
linked-to(x, y) if linked-to(y, x)
```

できる。

これは図14の結果を生じる。

処理は，導出定理証明のプロセスによって行われる。そこでホーン節は等価な内部形式へと変換され，結果が得られるか矛盾が生じるまで述語の「相殺」が行われる。後者の場合，問い合わせの応答は得られない。

Prologの非手続き的性質にもかかわらず導出のプロセスは逐次的であり，ホーン節はプログラム上の出現順に評価される。このことは「緊密な」再帰と否定的情報の表現に関して多くの問題を生じさせる。たとえば，ほとんどの Prologの処理系で，図15の 2 つの節はインタプリタを無限ループへと誘い込んでしまう。

Prologの否定は，述語が真であるとの確定の失敗として解釈される。したがって述語not(component - of (x, y))が真となるためには，述語component-of(x, y)を真とするすべての試みが失敗しなければならない。

ロバート・コワルスキは次のように述べている。「失敗による否定の実現によって，否定条件の成功または失敗が実行される文脈に危険なまでに依存するようになってしまった」。

Prologのこの問題は多くの研究機関で調べられていて，それらの研究開発が注目を集めている。最近の研究として，Prologの並列的実現であるParlogやPrologとLISPの要素，あるいはPrologと手続き型言語を合成するさまざまな試みがなされている。

日本の第 5 世代コンピュータプロジェクトが核言語として論理型言語を採用しているのは衆知だが，そこで使用されているのはPrologを拡張した逐次型論理ベース言語KL0，それをさらに拡張してオブジェクト指向の機能とマクロ機能を付加したESPと呼ばれる言語である。逐次形推論マシン PSI のオペレーティングシステム SIMPOSは，ESPによって記述されている。また，並列論理型言語GHCに環境やスケジューラといった拡張を施したのがKL1(核言語KL1-B)で，並列推論マシン PIMのオペレーティングシステム PIMOSは，KL1(システム記述言語KL1-U) で記述されている。

まとめ

急ぎ足でプログラミング言語発達の足跡を辿ってきたが，大雑把にいって手続き型，関数型，論理型の流れがあることが理解いただけたと思う。図16にプログラミング言語の系譜をまとめてみた。 3 つの大きな流れがあるといっても，たがいに緊密に影響を及ぼしあっているのがわかる。

手続き型言語の源はチューリングの仮想機械まで遡れよう。また関数型言語の源もチャーチのラムダ計算に求めることができるわけだが，両者がヒルベルトの決定問題に解を与えるために考えられた仕組みであることを思う

プログラミング用語
後判定反復文　制御文の一種で，do〜文のように継続条件を後ろで判断する文。

と，両者の関係が同一物の表裏のそれのようにも思える．とりわけFORTRANとLISPがアーキテクチャやデータ構造という計算機の実物の姿を反映していることを考えると，その感が深い．翻って論理型言語の源の一階述語論理は決定問題とは関係ないし，そのデータ構造やコントロール構造も計算機の姿を映してはいない．実現にはLISPの影響を受けているとはいえ，数理論理学の記法から生まれたプログラミング言語であるという意味であえて「プログラミング言語の鬼子」と名付けさせてもらった．

本文では触れることができなかったが，Algolから分離発展した言語にSimulaというものがある．現在流行のオブジェクト指向言語の祖先である．その直系でオブジェクト指向の思想を純粋な形で伝えているのがSmalltalkであるが，そこで培われた情報隠蔽やメッセージ送信，そして属性の継承などの考え方は，手続き型，関数型，論理型を問わずすべての言語に影響を与えている．機会があれば別稿でこれらについてあたってみたい．

これまで多くの言語が現れたように，これからも多くのものが開発されていくであろう．それらが少しずつでも計算環境を改善し，プログラミングを楽しいものにしてくれることを期待したい．最後にCOBOLについてまったく触れなかったが，その名誉のために世界でもっとも広く使用されている言語であるとだけ述べておきたい．

参考文献

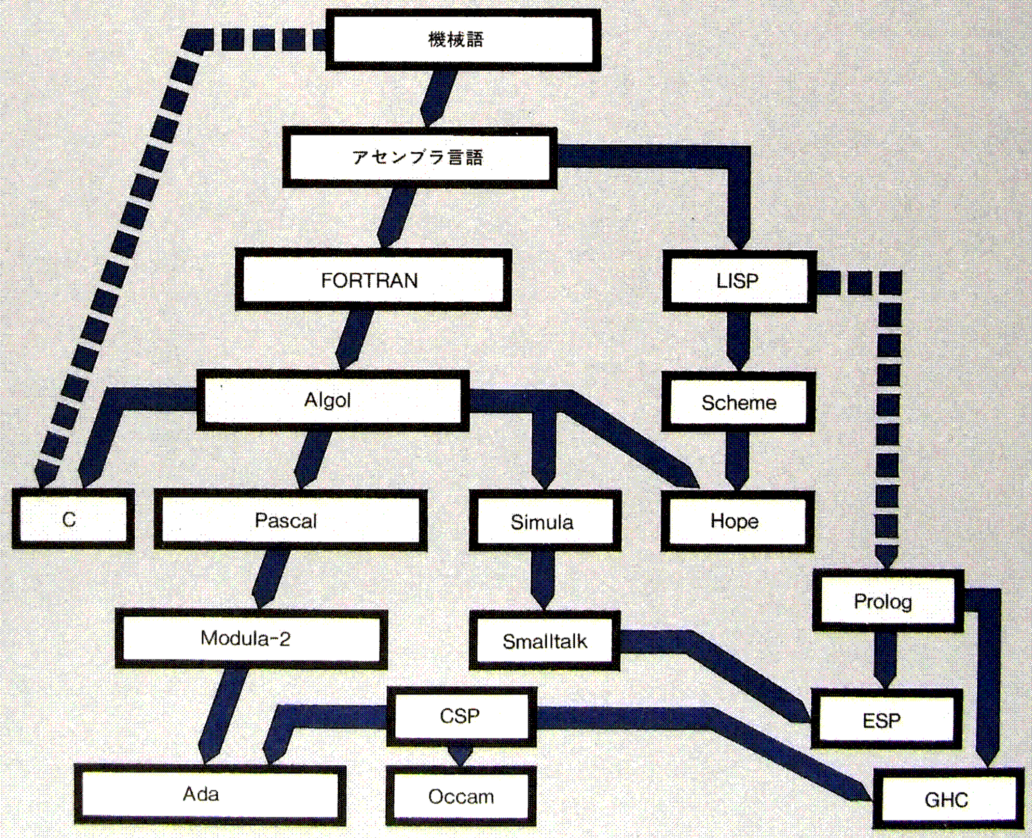
B. J. MacLennan, Principles of Programming Languages: Design, Evaluation, and Implementation, Holt, Rinehart and Winston, 1983

R. Sethi, Programming Languages: Concepts and Constructs, Addison-Wesley Publishing Company, 1989

萩谷昌巳,「ソフトウェア考現学」, CQ出版社, 1985

P. ビショップ, 神林靖訳,「新世代コンピュータと情報技術」, 啓学出版, 1990

図16　プログラミング言語の系譜



プログラミング用語
エスケープシーケンス　ソースプログラム作成時，文字列制御文字を記述するために使われる．0x1b(ESC)や0x5c('¥')に続く表記方法．

2

特集：この言語を使いなさい!!

私が勧めるこの言語

使い込んだ14人が14のプログラミング言語を解説

Assembler
BASIC
C
C++
COBOL
FORTRAN
HyperCard/HyperTalk
Lisp
LOGO
Mind
Modula-2
Pascal
Prolog
Smalltalk

TSR，デバドラなどに使われる DOSの主力言語

Assembler

小高輝真

アセンブリ言語とアセンブラ

本来「アセンブラ」とは，アセンブリ言語で記述されたソースコードをCPUが実行可能なバイナリコードに変換するプログラムのことを指す単語である．ところが，「アセンブリ言語」という単語が使われることはまれで，アセンブリ言語を指して「アセンブラ」と呼ぶことが多い．この傾向は英文の文献にも見られるようだ．個人的にはこのようなことにこだわるつもりはないのだが，本稿では両者を区別するため「アセンブリ言語」と「アセンブラ」を区別して用いることにする．

アセンブリ言語は，現在のMS-DOS上の開発言語としてC言語に次ぐ主力言語といえるだろう．アプリケーションプログラムの開発ではもっぱらC言語を補佐するために使われているが，デバイスドライバやTSRのようなプログラムではアセンブリ言語でのプログラミングが主流である．

MS-DOS上のアセンブラとしては，米マイクロソフト社のMASM(マクロアセンブラ）が早くからリファレンスとして存在している．現在は高速MASM互換製品として，米ボーランド社のTurbo Assemblerや豪SLRシステムズ社のOPTASMを選択することもできる．Turbo AssemblerはMASM互換モードに加え，独自の文法を用いる「アイデアルモード」ももっている．この文法は高級言語的で美しく，またMASMでは不可能な構造体のネストなどもできるのだが，MASMとの互換性の問題からあまり広く使われていないようだ．Turbo AssemblerとOPTASMは，条件分岐先が＋127，－128バイトを超えると自

プログラミング用語
エミュレーション　あるシステム上で別のシステムを模倣すること．数値演算プロセッサを持たないシステムでは数値演算エミュレータを使う．

リスト1　hello.c

```
/* K&R「プログラミング言語C」より */
main()
{
      printf("hello, world\n");
}
```

リスト2　hello.asm

```
;MASMの基本的機能のみを用いて記述した場合．
;MS-DOS 1.xx and later
;MASM 1.20 and later
;A>masm hello;
;A>link hello;
;A>exe2bin hello hello.com

code    SEGMENT
        ASSUME cs:code, ds:code, es:code
        ORG     100H

start:  mov     ah,09H
        mov     dx,OFFSET message
        int     21H
        ret

message DB 'hello, world',0DH,0AH,'$'

code    ENDS
END     start
```

表1　ソースとオブジェクトのサイズ比較

-a--w	67	Nov 17 1990 05:11	hello.c
-a--w	6610	Nov 17 1990 05:11	hello.exe
-a--w	312	Nov 17 1990 17:54	hello.asm
-a--w	23	Nov 17 1990 17:54	hello.com
-a--w	664	Nov 18 1990 18:49	bonjour.asm
-a--w	570	Nov 18 1990 18:49	bonjour.exe

＊ hello.cは，Turbo cのデフォルト設定でコンパイルした

リスト3　bonjour.asm

```
;MASMの各種機能を用い抽象化度を高めて記述した場合．
;MS-DOS 2.xx and later
;MASM 5.10 and later
;A>masm bonjour;
;A>link bonjour;

        DOSSEG
        .MODEL SMALL
        .STACK 100H

;/* 定数定義 */
CR = 0DH
LF = 0AH
STDIN  = 0
STDOUT = 1
STDERR = 2

;/* マクロ定義 */
puts    MACRO   message
        LOCAL   address,length
        .DATA
        address DB      message
        length  EQU     $ - OFFSET address
        .CODE
        mov     ah,40H
        mov     bx,STDOUT
        mov     cx,length
        mov     dx,OFFSET address
        int     21H
        ENDM

exit    MACRO   status
        mov     ah,4CH
        mov     al,status
        int     21H
        ENDM

;/* プログラム本体 */
        .CODE
main    PROC    NEAR
        mov     ax,@DATA
        mov     ds,ax

        puts    <"bonjour, chapeau",CR,LF>
        exit    0
main    ENDP

END     main
```

動的にジャンプ命令を構成しなおしてくれるような機能も備えている．なお，OPTASMはMASMバージョン5.0互換であるため，5.10から導入された多くの新しい機能が利用できないという欠点がある．

アセンブリ言語 VS C言語

アセンブリ言語とC言語で書いた具体的なプログラムを比較することで，アセンブリ言語の特徴について述べてみたい．

リスト1はC言語で書かれたプログラムとしてはもっとも有名な「hello, world」を出力するhello.c，リスト2はMASMの基本的な機能を用いて記述したhello.asm，リスト3はMASMの各種機能を用いて多少C風に記述したbonjour.asmである．hello.asmはMASM Ver.1.20以降で，bonjour.asmはVer.5.10以降でアセンブル可能である（余談になるが，PC-9801RA21でMASM1.20，1.27を実行しようとするとハングしてしまった．詳しい原因は不明だが，80386で実行するとハングすることがわかった．V30や80286では正常に実行できた）．表1には各ソースとその実行形式のサイズを示した．

可読性

まず，3種類のリストを見比べて感じるのは可読性の違いである．hello.cは「文字列を出力する」ということを端的に表している．それに対して，hello.asmではOSのシステムコールが見えているに過ぎない．また表示する文字列も離れた場所に記述されている．bonjour.asmではマクロの効果でmain部の可読性が高まっていること

を感じていただけると思う．しかし，もっと複雑な処理になると可読性の向上には限界があるし，マクロなどの定義は一からプログラマが行う必要があるので結構手間がかかる．

この例では表現しきれないが，アセンブリ言語では再帰のような複雑なアルゴリズムも記述しにくい．また，記述したとしても可読性の低さのために保守性が悪くなる．

可読性が高い，ということはすなわち生産性，保守性が高いということだ．生産効率を考える場合には明らかにC言語のほうが優れている．規模が大きくなるとこれはさらに顕著になる．

非ブラックボックス性

OSのシステムコールが見えているということは，細かな処理をする場合には重要なことになる．たとえば，ある関数がシステムコールを呼び出すのかどうか，ということは外部割込やデバイスドライバの処理ルーチンを書く場合には重要である．一般的には外部割込やデバイスドライバからシステムコールを呼ぶことはできないからだ．そのほか，再入可能かどうかも問題になる（フォアグラウンドと割込処理ルーチンの両方から同じ関数を呼び出すような場合）．

アセンブリ言語では，すべてのコードをプログラマが管理するので，ブラックボックスになる部分は存在しない．システム寄りのプログラムの記述には好都合である．

速度

プログラム本体の実行速度はhello.com，bonjour.exeが圧倒的に高速な「はず」だ．しかし，MS-DOSの文字列出力自体かなり遅いものなので，本体の速度は誤差の範疇に入ってしまう．この例ではアセンブリ言語によるメリットはない．あるとすれば，フロッピーディスクから起動するときの読み込み時間が短いことくらいだろう．速度を要求されない部分では，低い生産性でアセンブリ言語で記述しても十分な見返りが得られない．

反対に，外部割込ルーチンのように高速な処理を要求される部分，ひんぱんに呼び出されるBIOSやデバイスドライバ，精密なタイミングが必要なデバイス制御，アプリケーションプログラムでも速度の要求される部分などではアセンブリ言語の高速性が生きてくる．

サイズ

表１を見てみると，スタートアップルーチンや仕様の大きなprintfのコードを含むhello.exeは大きく，必要最小限のコードだけで構成されているhello.comは非常に小さい．bonjour.exeは，XENIX互換ファンクションを用いたためのサイズ増加もあるが，それは僅かでEXEモデルの再配置情報による増加分がほとんどである．

もちろん，実用的プログラムではここまでドラスティックな違いが表れることはない．それでも，同様の動作をするプログラムをC言語とアセンブリ言語で作れば，サイズの面ではアセンブリ言語のほうがつねにはるかに小さなサイズに仕上がる．

メインメモリ上に常駐するTSRやデバイスドライバでは，常駐サイズの縮小化が厳しく要求されるので，この点ではアセンブリ言語が非常に有利になる．

可搬性

アセンブリ言語には可搬性がない．一方，C言語にはCPU，OSを超えた可搬性がある．可搬性を求めるなら，アセンブリ言語は使えない．

アセンブリ言語の将来

アセンブリ言語は以上のような特徴をもっている．欠点も多いが，アセンブリ言語だけに与えられた長所も備えている「個性的な」言語だと思う．

しかし，プログラミング言語界のトレンドは生産効率と可搬性を求めている．この流れに乗り切れないアセンブリ言語は今後どうなるのだろうか．

UNIX系OSの世界では，可搬性が要求されるためアセンブリ言語が使われることはほとんどない．OS自体，C言語で書かれている（MS-DOSはアセンブリ言語）．また，CPUパワーの向上により，高級言語で十分な処理速度が得られる適用範囲は広がる一方だ．アーキテクチャの設計段階から人がアセンブリ言語のソースを書くことを想定していないRISC型CPUも一般化している．

パソコンのプログラム開発においても，アセンブリ言語のシェアは減少することはあっても増加することはないだろう．それでも，アセンブリ言語がほとんど不要になるまでにはまだ相当の時間がかかるはずだ．まだしばらくは，カリカリとアセンブリ言語でプログラミングする楽しみ（苦しみ？）を味わうことができそうだ．

プログラミング用語
オーバーヘッド　演算などが実行されない時間のこと．C言語では関数呼び出しは変数参照よりオーバーヘッドが大きい．

処 理 系 紹 介

Assembler

MASM 5.1

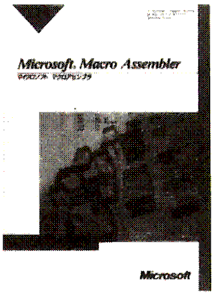
MASM 5.1 (Microsoft Macro Assembler Version 5.1)は，パソコン上の標準OSであるMS-DOSおよび将来の標準になるであろうといわれているMS OS/2を提供しているマイクロソフト社が発売しているアセンブラパッケージである．OSメーカーが提供するアセンブラということ，機能が豊富なこと，CodeViewと呼ばれる強力なソースレベルデバッガが付属していることなどから，現在のパソコンにおける標準アセンブラとしての地位を確立している．

MASM 5.1のおもな特徴として，MS OS/2に対応し，ファミリーアプリケーションインターフェースを利用したMS OS/2でもMS-DOSでも動作するアプリケーションの作成が可能で，さらにMS OS/2の direct OS call，ダイナミックリンク，マルチスレッドを利用したアプリケーションの作成も可能となっている点があげられる．

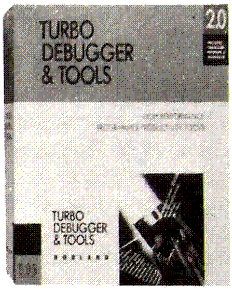
また，MS-DOSおよびMS OS/2のそれぞれで動作するソースレベルデバッガCodeView，多機能エディタMicrosoft Editorを標準装備し，MS OS/2においてはインクリメンタルリンカもサポートしている．

書式上の特徴としては，簡易化セグメントをサポートし，メモリモデルの記述が簡単に行える点があげられる．

MASM 5.1は，8086，80286，80386，8087，80287，80387用のオブジェクトコードを作成することができる．また，MASM 5.1で作成されたオブジェクトファイルはC言語やPascalなどの他の高級言語で作成されたオブジェクトファイルとリンクすることができるため，ミクスドランゲージプログラミングが可能となる．



製品名：Microsoft Macro Assembler 5.1
価格：40,000円
問い合わせ先：マイクロソフト（株）
03-3363-1201



製品名：Turbo Debugger & Tools 2.0
価格：39,800円
問い合わせ先：(株)ボーランドジャパン
03-3486-1400

Turbo Assembler

Turbo Assemblerは，米国Borland社が自社内で使用していた開発ツールをそのまま製品にしたといわれているMASMとほぼ完全な互換性をもった高速アセンブラである．

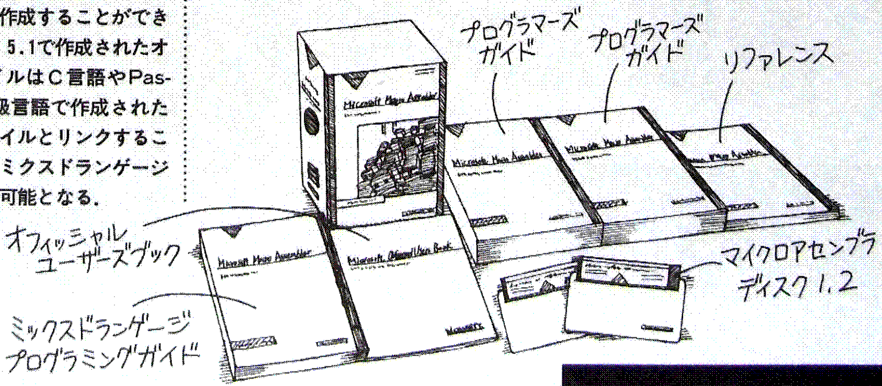
Borland社の他の言語製品であるTurbo CやTurbo Pascalのような統合環境こそ備えていないものの，MASMのほぼ3倍といわれる高速なアセンブル，80386の仮想86モードデバッグを実現したTurbo Debuggerやユーザーが作成したプログラムの実行効率を計測するツールであるTurbo Profilerを装備していることからその人気は高い．

Turbo Assemblerのもっとも大きな特徴は，Idealモードと呼ばれるMASMから派生したアセンブルモードを備えている点である．

Idealモードのすべての機能はMASMの機能を拡張または改良したもので，MASMの演算子やキーワードをより単純明快にし，厳密な型チェックを行うようにしたものといえる．

Idealモードを用いることで，従来ならば実行時またはデバッグ時でなければ発見することのできなかったバグをアセンブル時に発見することが可能となる．

Turbo Assemblerは，8086，80186，80286，80386，80486，8087，80287，80387のオブジェクトコードを作成することができる．また，Turbo C，Turbo Pascalなどと同時に使用することも可能である．



Illustration：Ruriko Masuda

プログラミング用語
オフセット　ある基準からの相対的な位置のこと．メモリやファイルの位置を示すのに使われる．

Assembler

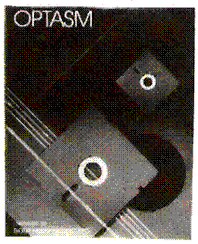
OPTASM

OPTASMは，米国SLR Systems社が発売している1分間に75000行（MASMの約5～10倍）という高速アセンブルを実現したMASM 5.0互換のアセンブラである．

8086系のCPUでは，セグメントの範囲外への条件ジャンプをする場合複数のジャンプ命令を組み合わせて使用する必要があるが，OPTASMはこのようなジャンプを自動的に処理してくれる．また，ラージモデルのアプリケーションを作成する場合には，同じセグメントにあるサブルーチンをファーコールで呼び出す必要があるが，OPTASMはファーコールをPUSHとCSとニアコールで置き換えるため高速にコードを実行することができるようになる．

また，コンフィギュレーションを行うユーティリティにより，MASMに合わせてバグや互換性，追加機能などを制御することが可能である．

OPTASMは，その信じられないほどのアセンブルスピードによって多くの支持を集めているが，MASM 5.1互換ではない，80386のインストラクションに対応していない，ミクストランゲージプログラミングをサポートしていないといった点で不満も残る．早急なバージョンアップが望まれるところだ．



製品名：SLR OPTASM
価格：19,800円
問い合わせ先：㈱サザンパシフィック
045-314-9514

マシン語マスター98

マシン語マスター98は，MS-DOS版N88-BASIC（86）の環境で使用できるアセンブラである．Z80のアセンブラに慣れた人やアセンブラの初心者を対象に作られており，リンカを必要とせず即実行可能なHEXファイルを作成するためROM化などに最適となっている．

エディタ，アセンブラ，デバッガがメモリ上に同居しているため，デバッグ後の再アセンブルなども高速に行うことが可能である．

マシン語マスター98は，V30，8086，80286に対応している．また，マシン語マスター98を実行するには，MS-DOS版N88-日本語BASIC（86）が必要である．

製品名：マシン語マスター98
価格：32,000円
問い合わせ先：暁商会
0138-51-7198

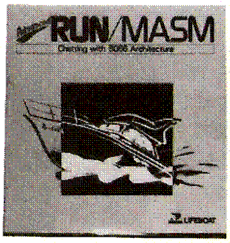
Advanced RUN/MASM

Advanced RUN/MASMは，8086/80286のアセンブラプログラムをメモリ上でアセンブルし，即座に実行するオンメモリアセンブラである．

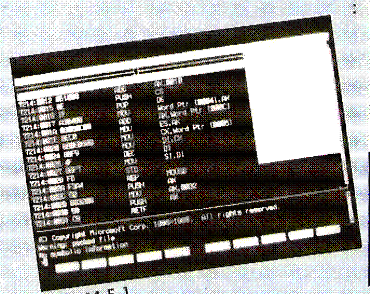
BASICライクなインターフェースとエディタを内蔵し，SEGMENTやASSUMEなどの複雑な疑似命令を使用することなく命令のみを記述することができるため，アセンブラの学習に最適なシステムになっている．

また，Advanced RUN/MASMは直接COMファイルを出力することができるため，リンクおよびCOMファイルへの変換操作を必要としない．OBJファイルを出力することも可能である．

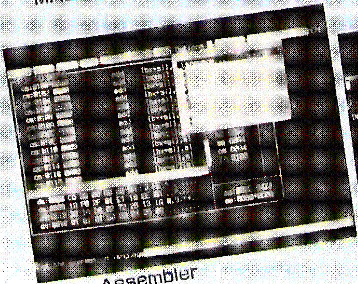
Advanced RUN/MASMのニーモニックや疑似命令は，MASMに準拠している．



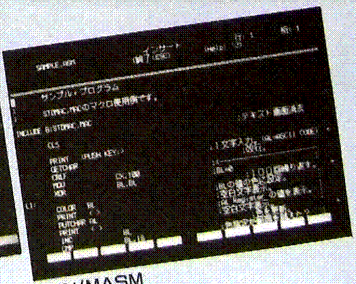
製品名：Advanced RUN/MASM
価格：29,800円
問い合わせ先：㈱ライフボート
03-3293-4711



MASM 5.1



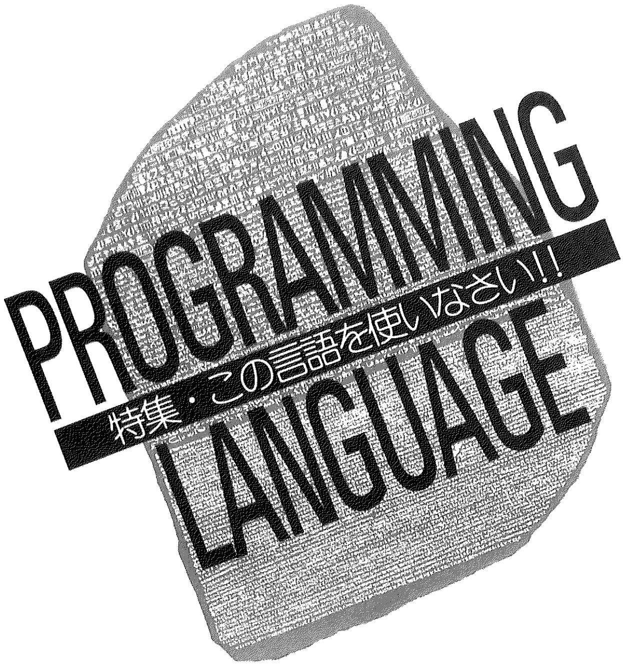
Turbo Assembler



RUN/MASM

処　理　系　紹　介

プログラミング用語
オブジェクト指向　オブジェクトに着目し，カプセル化・継承・多態性などによって各要素を抽象化するプログラミング手法



すぐに乗れる自転車のような手軽さ

BASIC　大阪府科学教育センター情報工学教室　山本英昭

情報教育に活用して……

私は，大阪府の小・中・高校および養護教育諸学校の先生方を対象とした研修・研究センターに勤めており，そこの教育工学および情報教育関係を担当している教室で仕事をしている．本センターでは，現在，情報教育関係の研修を実施しており，年間30日間の長期の研修，10日間の中期の研修，3～4日の短期の研修をそれぞれいくつか開講している．

長期の研修は主として汎用機を用いて，FORTRAN, COBOLによるプログラミング，アプリケーションの利用およびその応用などを行い，中期・短期の研修は主としてパソコンを用いて，BASICによるプログラミングおよびアプリケーションの利用などを行っている．BASICは，研修に参加される先生方の学校に設置されているパソコンの機種がさまざまであるため，できるかぎり互換性があるようにしたいということから，BASIC/98を用いている．

とっつきのよさが魅力

BASICはその名前のBeginner's All－purpose Symbolic Instruction Codeが示す通り，初心者教育用の多目的対話型処理用言語として1960年代半ばにアメリカで開発された．当初は汎用機のTSS（タイム・シェアリング・システム）用言語として，生徒の学習用に使用されていたが，1970年代半ばにパソコン用のBASICインタプリタが考案されて以来，パソコンの主要言語として普及してきた．

日本では，1982年にJIS C 6207 Minimal BASICが制定されたが，パソコンの急激な普及とともに発達し，各メーカーやソフトハウスなどがそれぞれ独自の機能を付け加えて，現在多数の種類が出回っている．

BASICのおもな特徴は，インタプリタ方式であること，コマンドモードとプログラムモードの2つのモードをもつことであり，最大の魅力は，日本語処理，グラフィック処理，ミュージック処理が使いやすく，厳しい言語仕様ではないため，まったくの初心者でも簡単に勉強がはじめられるとっつきのよさであると思う．

チームを組んでそれぞれ分担しながら市販用のあるいは業務用のアプリケーションやゲームなどの開発にあたる職業的なプログラマは別として，自分の仕事にちょっとパソコンを利用したりあるいは趣味でプログラムの勉強をはじめる人にはもっとも適したプログラム言語のひとつであるといえる．

コンピュータ言語は，大きくコンパイラ方式とインタプリタ方式に分けることができる．コンパイラ方式は，一般には，エディタによるソースプログラムの作成，コンパイル，リンク，実行という手続きを踏む必要があり，しかも各手続きは，それぞれ別のシステムプログラムを起動させなければならない．これが初心者には相当複雑な操作であり，理解が困難であるにもかかわらず，この手続きの理解，マスターがなければ，プログラムの実行結果を求めることができない．このようなとっつきの悪さと，さらに，プログラムのミスや操作に誤りがあった場合には，この手続きをやり直さなければならないという煩雑さがある．

これに反して，インタプリタ方式は，プログラムを1行ずつ読み取り，その意味を解釈しながらプログラムを逐次実行していく方式である．コンパイラ方式に比べて，実行速度が遅いという欠点をもつが，プログラムが未完成であっても，あるいはミスがあっても途中まで実行させることができたり，途中から実行させることができるという簡便さがある．その上，BASIC/98ではプログラムを1行入力するごとに文法のチェックをしてくれるので，初心者にとっては非常に親切な言語であるといえる．また，多くのBASICはコンパイルできるようになっているので，プログラムが完成して正常に実行できるようになってからコンパイルして使用することも可能である．

入力ミスを自分自身の目で確認

本センターの研修では，BASICのプログラミングの導入に図1のようなプログラムを用いているが，プログラムを1行入力するごとに実行するようにすれば，いちいち細かな説明がなくてもその行の命令の意味が理解できる．

たとえば，LINE文でパラメータをまちがえて入力したりすれば，線の位置などが違って表示されるので，入力ミスを自分の目で確認し，自分自身でまちがいがチェックができるので，プログラムの学習をするのに非常に適している．とくに，200行，220行のPAINT文では，よく画面全部を塗りつぶしてしまうというトラブルが起きるが，その直前にSTOP文を入れることによって，PAINT文の直前まで実行させ，トラブルの原因を検討するということが簡単にできることも非常にありがたい．

コマンドモードとプログラムモードの両方が使えることによって，たとえば，文法上ミスがなくても論理的なミス等があって，正常な結果が得られない場合，プログラム実行後あるいは途中まで実行させた状態で，メモリの中身や変数の値を，PRINT文を直接実行することで簡単に確認できる．このことは，プログラムのデバッグを容易にするとともに，研修の先生方のプロ

図1　プログラムリスト

```
100 CLS 3
110 LINE(100,300)-(500,300),2
120 LINE(200,300)-(200,200),5
130 LINE(400,300)-(400,200),5
140 LINE(150,200)-(450,200),4
150 LINE(150,200)-(300, 80),4
160 LINE(450,200)-(300, 80),4
170 LINE(220,220)-(250,280),7,BF
180 LINE(325,250)-(375,220),1,B
190 LINE(340,150)-(360, 95),4,B
200 PAINT(300,150),5,4
210 CIRCLE(100,50),30,6,0,6.28318,1,F
220 PAINT(100,50),6
230 CIRCLE(350,75),10,7,,,,F
240 CIRCLE(400,50),14,7,,,,7,F
250 CIRCLE(470,30),20,7,,,,5,F
260 PSET(550,20),6
270 PSET(550,19),6
280 PSET(549,20),6
290 PSET(550,21),6
```

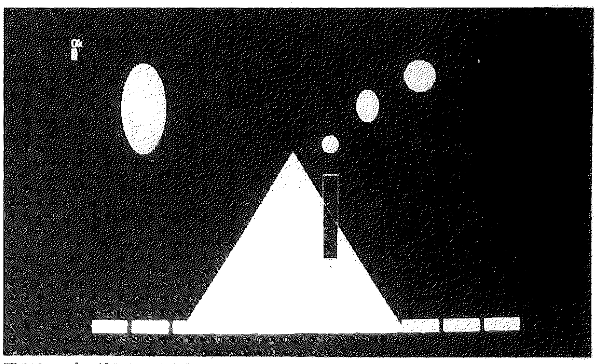


写真1　プログラムの実行画面

プログラミング用語
型宣言　各種変数を利用するときコンパイラに対して宣言をする．宣言しなくても自動的に型を判断して変数を用意してくれる処理系もある．

グラムのミスを発見し指摘する際たいへん便利である.

とりあえずプログラムを組めることが大切

BASICはその言語仕様のゆるさとインタプリタの操作性のよさのために，試行錯誤的にプログラムを開発してしまう．つまり，きちんとプログラムの設計をせずにいきなりパソコンに向って入力をはじめてしまい，誤りがあれば，そのつどいきあたりばったりで修正をしていくからプログラムに一貫性がなく，わかりにくいプログラムになってしまうという批判もある.

しかし，逆に，前もってこれらのことを勉強しておかないとプログラムが組めないならば，初心者にとってプログラムを組むということは，そのスタートにおいて非常に高いハードルを越えなければならない．プログラムを組むその第１歩目であきらめさせてしまうことは，高度情報化時代の新しい資質の育成の障害となるように思う．けっしてプログラムのわかりやすさや構造化がどうでもよいというわけではないが，実際にプログラムを組むことができ，そのプログラムをきちんと実行させることができてはじめて仕事や問題が解決するのであって，このステップを越えることができなければ，単なるペーパードライバーと同じことではないかと思う.

そして，情報教育を担当する先生方や，プログラムに興味を示す人には，とりあえずプログラムを実際に組むことができ，実際に実行させることができるようになってから，アルゴリズムやプログラムの構造化といったわかりやすいプログラムを組むことを学習していただくことが大切であると考えている.

最後に，繰り返しになるが，職業的なプログラマは別として，一般の多くのプログラマは，専門の仕事は市販のアプリケーションソフトを用い，簡単な事務計算や数値計算などは自分用のプログラムを組むといったスタイルが普通ではないかと思う．このスタイルには，相当時間をかけて練習しなければ乗れない自動車ではなく，ちょっと練習すればすぐに乗れる自転車のように，手軽に利用できるBASICという言語がもっとも適しているのではないかと思っている.

APL

APL (A Programming Language) は1962年，米ハーバード大学のK.E.Iverson博士の手によって数学における各種計算記号や手順などの記述規約統一のために考案された言語体系で，考案された当時はコンピュータ上で動かすことを目的にしてはいなかった.

1964年，K.E.Iverson博士はIBM社の研究所に移り，このAPLの言語体系をIBM360上でAPL/360として完成させた．APLそのものはコンピュータ上では実現できないので，当初設計されたAPLからは変更されている．このAPL/360は数学演算記述言語のためデータファイルの概念をもたず，キーボードから入力された計算記号や手順を処理し，結果をディスプレイに出力することしかできなかった．この問題を解決したのがAPLSVである．データファイルの概念をもつことによって他の言語と同等のコミュニケーションをAPLは得たのである.

APLの最大の特徴は複雑な配列変数も単純変数と同様に簡単な指定で処理できることにある．特定の値を配列の全要素から検索したり，並び替えたり，削除したりなど，APLは膨大な数値データを扱うデータベース言語として向いている.

APLの最大の欠点は，変数のためのメモリが大量に必要であったこと，コンピュータが扱う文字コードには含まれていない独自のコードを必要とし，通常のコンピュータ上で実現することが難しいことである．APLをコンピュータ上で稼働させる場合は専用のキャラクタROMを搭載するか，表示速度を犠牲にしてビットマップで表示する必要がある．このことがAPLの普及を妨げる決定的な要因となった.

処理系紹介

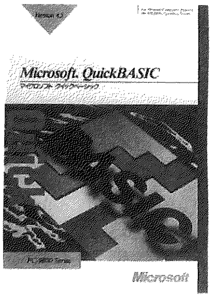
BASIC

MS-QuickBASIC

米Microsoft開発のBASICコンパイラ&インタプリタ．エディタを中心とする統合環境でプログラム作成，インタプリタによる実行，コンパイル，リンク，デバッグとプログラム開発に要するすべての処理を実行することができる．QBアドバイザと呼ばれる強力なオンラインヘルプ機能を備え，関数の説明やエラーメッセージの内容などが参照できる．このヘルプはハイパーテキスト風に次々と必要な項目についての説明を表示するようになっている．デバッグ機能もブレークポイントの設定，変数のウォッチ，トレース等ひととおり充実している．

言語仕様としては構造化のための制御文をサポートし，モジュール化プログラミングの可能な拡張BASICとなっている．新しいBASICの概念と統合環境の操作方法を学習するための学習ソフトQBチュートリアルが付属．インターフェース，マニュアルの説明，価格設定などどれをとっても一見初心者向けだが，かなり複雑なアプリケーションを構築できるパフォーマンスをもつ．

NEC PC-9800シリーズ対応（ただしXA/LTとハイレゾモードを除く）．

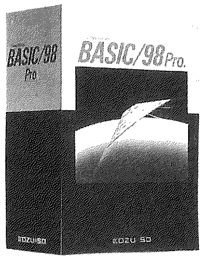


製品名：Microsoft QuickBASIC Ver.4.5
価格：20,000円
問い合わせ先：マイクロソフト㈱
03-3363-1201

BASIC/98Pro

NEC PC-9800シリーズに標準添付されているN88-BASICとの互換性を考慮したBASICインタプリタ＆コンパイラ．構造化プログラミングができるように一部仕様拡張されている．インタプリタ版は専用エディタを内蔵．コンパイラ版はインクリメンタルコンパイル方式を採用し，高速．また作業領域をEMS上に取ることも可能．

NEC PC-9800シリーズ，EPSON PC-286シリーズ，富士通FMRシリーズ，松下Panacom Mシリーズ，東芝J-3100シリーズに対応．

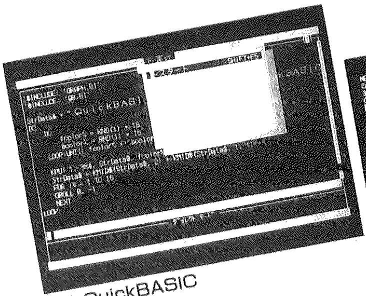


製品名：BASIC/98Pro インタプリタ
価格：27,000円
製品名：BASIC/98Pro コンパイラ
価格：57,000円
問い合わせ先：コーツシステムプロダクツ㈱
03-3232-1361

N88-BASIC

かつて8bitパソコンが主流であった時代には，BASICが内蔵されているのが普通であった．16bitのMS-DOSパソコンが全盛の今もその流れを受け継いでいるのが日本電気のN88-BASICである．PC-9800シリーズにはROMとして内蔵されているほか，ディスクアクセス可能なディスク版N88-BASICが標準で添付されている．このほかにMS-DOS上で動作するMS-DOS版やコンパイラも販売されている．

製品名：N88-日本語BASIC（86）（MS-DOS版）Ver.6.0
価格：16,000円または17,000円（メディアによって異なる）
製品名：N88-日本語BASIC（86）コンパイラ Ver.6.
価格：46,000円または47,000円（メディアによって異なる）
問い合わせ先：日本電気㈱パソコンインフォメーションセンター
03-3452-8000


MS-QuickBASIC

N88-BASIC

プログラミング用語
記憶クラス　変数をメモリエリアに保持する期間や，結合状態のこと．

PROGRAMMING LANGUAGE

特集・この言語を使いなさい!!

考えたことを素直に記述できる

C　　　　辻村　浩

C

私が勧めるこの言語

自由度の高いコーディングが魅力

C言語は私にとってもっとも身近なプログラミング言語であり，ほとんど母国語といってもいいくらい，ひんぱんに使っている言語です．Cのプログラムを書かない日は1日もないくらいですから．私の仕事はUNIXのカーネルをいじることなので，当然といえば当然なんですけどね．

私がなぜC言語を好むかというと，自由度の高いコーディングができるからです．自由度の高いコーディングというのは，自分の考えた通りのことを素直に記述できるということで，プログラムを組む人間にとっては非常に重要な要素です．また，C言語はポインタを使ってメモリやI/Oポートを直接さわったり，構造体や共用体を使ってデータの構造化を行ったり，非常に低級な（ハードウェアに近い）ことから，高級な（人間の発想の仕方に近い）ことまで，幅広く記述できます．

C言語をマスターするにあたっての最大の難関といわれているポインタの概念が，C言語の特徴をよく物語っています．ポインタは日本語に訳すと「指し示すもの」となりますが，データの格納されている場所（アドレス）を指し示すのに使われます．ポインタを使うことによって，構造化されたデータを効率よくアクセスするプログラムを書くことができます．また，ポインタはデータに限らず，メモリ中の特定のアドレスを参照したり，メモリにマップされているI/Oレジスタをいじるのにも利用されます．Cがデバイスドライバの記述から一般のアプリケーションの記述まで幅広く使われるのは，ポインタが利用できるからといっても過

プログラミング用語
疑似命令　コンパイル時などに，目的ファイルへ変換されるのではなく，処理系に対し指令を行う命令を差す．

者ではないでしょう．

ゲームのために開発された言語？

C言語は「UNIXという米国AT&T社が開発したミニコン用のオペレーティングシステムを記述するために設計された言語」であるといわれていますが，ほんとうのところはどうなのでしょう？　私はUNIX上でゲームをつくって遊ぶために開発された言語だと思っています:-)．

C言語はUNIX上で開発され，UNIX自体がC言語で書き直されました．その結果，移植性の高くなったUNIXが次第に普及し，それと同時にC言語も普及してきました．やがて，パーソナルコンピュータが出現し，MS-DOSが登場するや，C言語はあっという間にパソコン用の主要プログラミング言語のひとつになってしまったというわけです．

このような背景があるため，C言語はシステム記述向きといわれていますが，最近はいろいろな分野のアプリケーションを記述するのにも使われています．たとえば，あの「一太郎」もC言語で書かれているといわれています．Turbo CやQuick Cなど，比較的廉価なCコンパイラが市販されるようになり，Cプログラマの人口が増え，Cで書かれたプログラムもどんどん増えています．

言語というよりはツール

流行に遅れまいとC言語の勉強をはじめた人もいると思いますが，C言語はPascalなどのプログラミング教育用に開発された言語とは違って，かなり癖のある言語です．ある程度プログラミングの経験を積んだ人でないと，完全にマスターするのはむずかしいでしょう．

しかし，いったんマスターしてしまえば，強力なプログラム開発ツール（Cは言語というよりツールなんですね，少なくともカーネル屋にとっては）を手にしたも同然です．プログラムを組むのが好きでたまらない人にはぜひともC言語をマスターして欲しいです．今をときめくオブジェクト指向言語に走る前にC言語をものにしてみませんか？　ほかの言語を勉強するにしても，C言語を知っているのと知らないのとでは大違いです．

サンプルプログラム

では，最後に簡単なプログラムを紹介します．処理系依存の関数は使用していませんので，ANSI C対応コンパイラであればどれでもコンパイルできると思います．Turbo C 2.0 を用いて動作確認しました．

サンプルプログラム

```
/*
 *      今日はどういう日か表示する
 *      Copyright (c) 1990 by H.Tsujimura(pcs30332, tsujimura@otsl.oki.co.jp)
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

const char      *hoshi[] =
{ "九紫", "八白", "七赤", "六白", "五黄", "四緑", "三碧", "二黒", "一白" };
const char      *kan[] =
{ "甲", "乙", "丙", "丁", "戊", "己", "庚", "辛", "壬", "癸" };
const char      *shi[] =
{ "子", "丑", "寅", "卯", "辰", "巳", "午", "未", "申", "酉", "戌", "亥" };
const char      *yoh[] =
{ "日", "月", "火", "水", "木", "金", "土" };
const char      *tsuki[] =
{ "睦月", "如月", "弥生", "卯月", "皐月", "水無月", "文月", "葉月", "長月",
  "神無月", "霜月", "師走" };

#define JULIAN  1720994.0
#define MONTH   30.6001
#define YEAR    365.25

double  julian( int year, int month, int day )
{
        int     yy, mm, a, b;
        double  c, d;
        char    tmp[ BUFSIZ ];

        if ( month > 2 ) { yy = year;     mm = month;      }
        else             { yy = year - 1; mm = month + 12; }
        a = yy / 100;                           b = 2 - a + ( a / 4 );
        sprintf( tmp, "%f", YEAR * yy );        c = atol( tmp ) * 1.0;
        sprintf( tmp, "%f", MONTH * ( mm + 1 ) );   d = atol( tmp ) * 1.0;
        return ( JULIAN + d + c + ( b + day ) * 1.0 );
}

void    main()
{
        struct tm       *t = NULL;
        time_t          ctime;
        int             year, month, day, kyear, syear, kday, sday, hday;
        double          j;
        char            tmp[ BUFSIZ ];

        time( &ctime );  t = localtime( &ctime );
        day = t->tm_mday;  month = t->tm_mon + 1;  year = t->tm_year + 1900;
        j = julian( year, month, day ); /* 通算ユリウス日 */
        kyear = ( year - 4 ) % 10;  syear = ( year - 4 ) % 12;
        sprintf( tmp, "%f", ( j + 2.0 ) / 9.0 );
        sprintf( tmp, "%f", ( j + 2.0 ) - 9.0 * atol( tmp ) );
        hday = atoi( tmp );
        sprintf( tmp, "%f", j / 10.0 );
        sprintf( tmp, "%f", j - 10.0 * atol( tmp ) );
        kday = atoi( tmp );
        sprintf( tmp, "%f", ( j + 2.0 ) / 12.0 );
        sprintf( tmp, "%f", ( j + 2.0 ) - 12.0 * atol( tmp ) );
        sday = atoi( tmp );

        printf( "今日は %s%s年%s%s%s日 (%s曜日) [%s] です¥n",
                kan[ kyear ], shi[ syear ], tsuki[ month - 1 ],
                kan[ kday ],  shi[ sday ],  yoh[ t->tm_wday ], hoshi[ hday ] );
}
```


プログラミング用語
共用体　複数のメンバーが集合したデータ構造で，共用体の各メンバーが同じアドレスを参照する．

処理系紹介

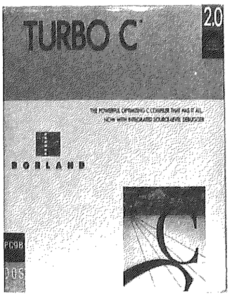
C

Turbo C

Turbo Pascalで成功した米Borland International社のCコンパイラ．エディタでソースプログラムを編集したその場でメニューからコンパイル，リンクを指定して実行し，デバッグする，いわゆる「統合環境」型のユーザーインターフェースが添付されている．コンパイル，リンクが高速，機種対応のグラフィックライブラリを備えている上，価格が安い．現在ではMicrosoft Cと並んで標準のCコンパイラとなっている．同社のTurbo DebuggerとTurbo Assemblerとの組合せで，強力な開発環境を提供する．

以前はTurbo C，Turbo DebuggerとTurbo AssemblerをセットにしたTurbo C Professionalというパッケージがあったが，現在はTurbo C 2.0単体の販売のみ．最近発売されたTurbo C++に，ANSI Cコンパイラとしての機能が受け継がれている．

NEC PC-9800シリーズ対応版，IBM PC対応版がある．



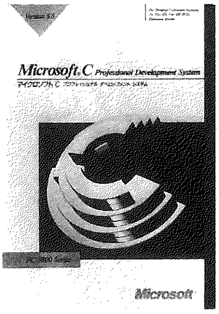
製品名：Turbo C 2.0
価格：19,500円
問い合わせ先：㈱ボーランドジャパン
03-3486-1400

MS-C

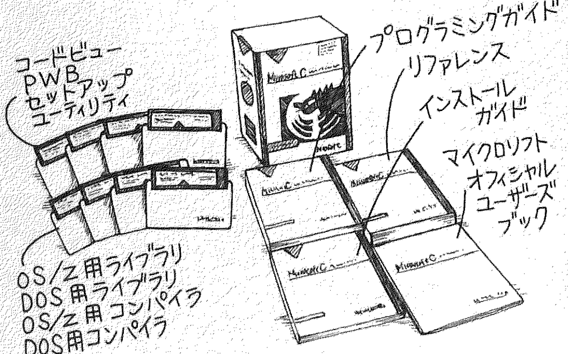
MS-DOSの開発元である米マイクロソフト社のCコンパイラ．MS-DOS上で動作するCコンパイラとしては事実上の標準となっている．最新のVer.6.0ではProgrammer's WorkBench (PWB) という統合開発環境とソースコードレベルデバッガCodeView Ver.3.0を標準装備．さらにインラインアセンブリ機能の追加，コンパイルスピードの速いクイックコンパイラの添付などいっそう機能強化されている．

Small, Medium, Compact, Large, Huge, Tinyの6種類のメモリモデルをサポートし，OS/2やMS-Windows上のアプリケーション開発環境をも提供する．そのためインストールには最低でもハードディスクに約6MB程度の空き容量が必要．フルセットでは20MBを要する(ただし，MS-DOSのみの開発環境なら12.5MB)．

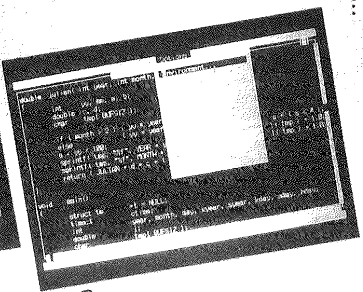
1990年12月末現在で日本語版のVer.6.0が出荷されているのはPC-9800シリーズ対応版のみであるが，富士通FMRシリーズ対応版や東芝J-3100シリーズ対応版等各社のパソコンに対応した製品が出荷される予定．



製品名：Microsoft C Professional Development System Ver.6.0
価格：98,000円
問い合わせ先：マイクロソフト㈱
03-3363-1201



Turbo C

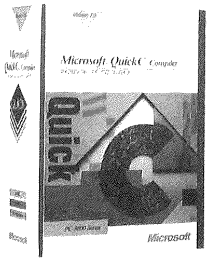


MS-C

C

MS-QuickC

米Microsoft社のCコンパイラ．エディタを中心とする統合環境でのプログラム作成，コンパイル，リンク，デバッグと，コマンドラインからの高速コンパイルという両方の環境をもっている．Quick Cの統合環境版にはQCアドバイザと呼ばれる強力なオンラインヘルプ機能があって，ハイパーテキスト風に次々と必要な項目についてのヘルプを表示できるようになっている．デバッグ機能もブレークポイントの設定，変数のウォッチ，トレース等ひととおり充実している．インターフェース，マニュアル，価格などの面から見るとかなり初心者を意識した作りであるが，機能的には同社のMicrosoft C Ver.5.1に優る部分もあって上級者の使用にも十分に耐えうる．



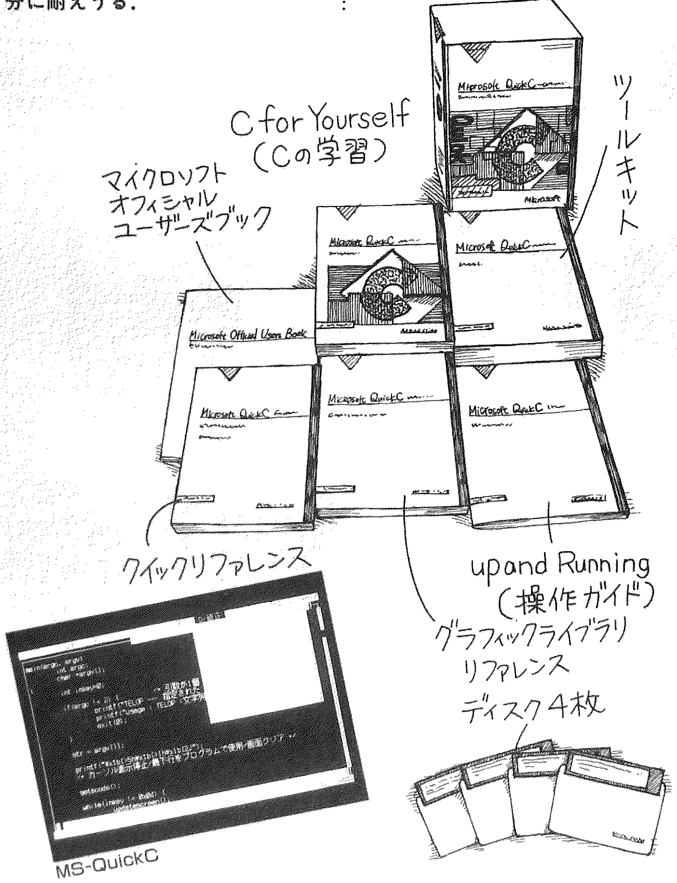
製品名：Microsoft QuickC Compiler Ver.2.0
価格：20,000円
問い合わせ先：マイクロソフト(株)
03-3363-1201

Lattice C

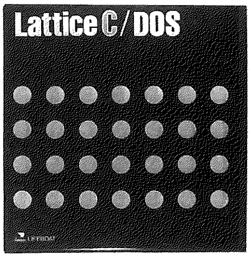
米Lattice社のCコンパイラ．かつてはマイクロソフト社がOEM採用していたこともあり，Cの処理系としての歴史は古い．ANSI規格に準拠．さらにUNIXSystemV上の標準ライブラリと互換性のあるライブラリを装備している．日本語によるコンパイルエラーメッセージの表示機能や，関数のプロトタイプ宣言の自動生成機能がある．

現在はDOS版のLattice C/DOSのほかにOS/2，MS-DOSの両方のアプリケーション作成が可能なLattice C/DUALやファームウェア開発用のLattice C/ROMable，リアルタイムマルチタスクシステム構築用のLattice C/RT，80386CPU対応のLattice C V86/RTなどの製品がある．

NEC PC-9800シリーズ，富士通FMRシリーズ，松下Panacom Mシリーズ，沖if800RX/EX，IBM PS/55シリーズ，東芝J-3100シリーズなど各社のパソコンに対応（ただし，Lattice C V86/RTはNEC PC-9800，FC-9800シリーズの80386CPU搭載機種に限る）．



MS-QuickC



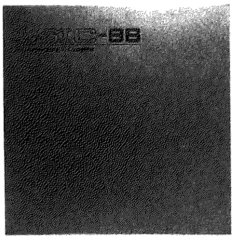
製品名：Lattice C/DOS Ver.4.1
価格：98,000円
製品名：Lattice C/DUAL Ver.4.1
価格：128,000円
製品名：Lattice C/ROMable Ver.4.1
価格：198,000円
製品名：Lattice C/RT Ver.4.3
価格：238,000円
製品名：Lattice C V86/RT Ver.4.3
価格：298,000円
問い合わせ先：(株)ライフボート
03-3293-4711

プログラミング用語
高級言語　CPUの構造に束縛されない言語．アセンブル言語のようにCPUよりでなく，より自然の言語に近い．

LSI C-86

インテル8086プロセッサ用Cコンパイラ．Cコンパイラとライブラリの他にROM化ツール，アセンブラ(r86)を含み，アセンブリ言語をCソースプログラム中に記述することができる(インラインアセンブラ機能) などROMプログラム開発を意識した構成になっている．ライブラリはUNIX，ANSIとコンパチブル．K&R第２版(Appendix)で定義された関数が含まれる．

Cコンパイラ，アセンブラ(r86)，ライブラリ，ライブラリのソース，ROM化ツール，トレーニングプログラム，サンプルプログラム，メイクなどのツール，各種ユーティリティを含む．



製品名：LSI C-86
価格：48,000円
問い合わせ先：エル・エス・アイジャパン　03-3404-1341

Whitesmith C

米Whitesmith社開発のCコンパイラ．同社のCコンパイラには各種ホスト用Cコンパイラと対象マシンを限定したクロスコンパイラがあり，マシン依存部分と非依存部分が明確に分けられているのが特徴．

パソコンで使用できるものには，インテル8086CPU用，日本電気V20/V30用がある．ANSI Cに準拠．ANSIライブラリ，拡張ANSIライブラリ，WSL独自ライブラリの３つのライブラリを組み合わせて使用できる．ROM化にも標準で対応．ソースファイル中のデバッグ領域のプリプロセッサによるON/OFF，同一ファイル中の関数やデータを異なったメモリモデルに割り当てるなど独自のデバッグ機能をもつ．

製品名：Whitesmith,Ltd. C 86 コンパイラ Ver.3.25
価格：500,000円
製品名：Whitesmith,Ltd. C V20/30 コンパイラ Ver.3.25
価格：500,000円
問い合わせ先：㈱アドバンスト・データ・コントロールス　03-3576-5351

NDP C 386

Microway社開発の80386用Cコンパイラ．80386CPUのプロテクトモードで動作するコードを生成する．実際のプログラム開発には別売のPharLap社の386｜TOOLBOX(DOS｜EXTENDER，プロテクトモード対応アセンブラ，リンカ，ライブラリアン，デバッガを含む)が必要となる．インラインアセンブラの使用，NDP FORTRAN386とのリンクが可能．

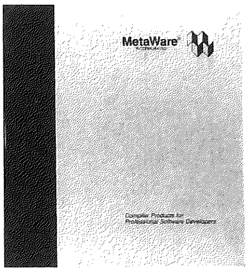
NEC PC-9801シリーズの32bit機，IBM PCとその互換機に対応．最低2MBの空き領域のあるハードディスクと640KB+2MB以上のプロテクトモード対応メモリが必要．数値演算プロセッサとして80287，80387，mW1167，mW3167のいずれかが必須である．

製品名：NDP C 386
価格：148,000円
製品名：386｜TOOLBOX Ver.2.2
価格：148,000円
問い合わせ先：テックソフトアンドサービス（株）　0474-23-1322

HIGH C386

MetaWare社の80386CPUのプロテクトモード用Cコンパイラ．386｜DOS-EXTENDERとの併用によりプロテクトモードで動作するプログラムの開発が可能．実際のプログラム開発には別売のPharLap社の386｜TOOLBOX（DOS｜EXTENDER，プロテクトモード対応アセンブラ，リンカ，ライブラリアン，デバッガを含む)が必要となる．

NEC PC-9800シリーズの32bit機，EPSON PC-386シリーズ，東芝J-3100シリーズの32bit機およびJ-3300，IBM PC/AT互換機(80386，80386SX搭載機)に対応．最低5MBの空き領域のあるハードディスクと1MB以上のプロテクトモード対応メモリが必要．



製品名：MetaWare HIGH C386
価格：148,000円
製品名：386｜TOOLBOX Ver.2.2
価格：148,000円
問い合わせ先：テックソフトアンドサービス（株）　0474-23-1322

Power C

ANSI C準拠のCコンパイラ．全ライブラリのソースファイルと，ソースレベルデバッガPowerCtraceが付属．NEC PC-9800シリーズとEPSON PC-286シリーズで動作する．

製品名：Power C compiler Ver.2.0.0J PC98版
価格：14,000円
問い合わせ先：システム・ワン㈱　03-3663-4551

プログラミング用語
構造化　プログラムの制御が上位のモジュールから下位のモジュールに渡される階層構造をとるプログラミング手法

C

Advanced RUN/C

ライフボート社と米Age of Reason社によって開発されたCインタプリタ.プログラミング言語の学習環境を提供することを目的としたライフボート社のAdvanced RUNシリーズのひとつ.

インターフェースはBASICライクなコマンドベース,日本語フルスクリーンエディタを内蔵,エラーメッセージの日本語表示など言語の学習を意識した機能をもつ.またLattice Cのライブラリ関数を取り込んでカスタマイズできる,作成したソースプログラムはLattice Cと互換性をもつ等インタプリタ→コンパイラへのステップアップも考慮されている.

NEC PC-9801シリーズ,富士通FMR50/60/70, FM TOWNS,東芝J-3100シリーズ,IBM PS/55シリーズに対応.

製品名：Advanced RUN/C
価格：29,800円
問い合わせ先：㈱ライフボート
03-3293-4711

C-terp

米Gimple Software開発のCインタプリタ.C-terpで作成したプログラムをMS-Cコンパイラにかけたり,MS-Cで作成した関数をC-terpの外部関数として取り込み,C-terp内部から使用することができる等,コンパイラと併用して利用することを前提に設計されている.

WORDSTARライクなエディタを内蔵.デバッグには対話型とトレースの2種類がある.対話型ではブレークポイント位置で,各種デバッグコマンドを入力.トレースはソースプログラム中にトレース用関数を埋め込むことによって,画面やファイルにトレースを出力させることができる.

NEC PC-9800シリーズ専用版,IBM-PC専用版,ANSI画面制御シーケンスを採用したANSI対応版がある.

製品名：C-terp MS-C用
価格：39,800円
問い合わせ先：㈱サザンパシフィック
045-314-9514

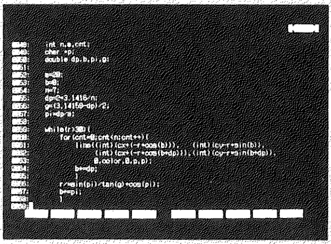
Let's C

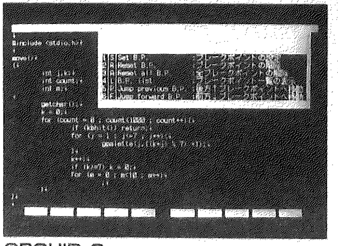
Cコンパイラ,デバッガ,エディタ(ソース付)のセット.NEC PC-9800シリーズ用とその他各種MS-DOSマシン用がある.

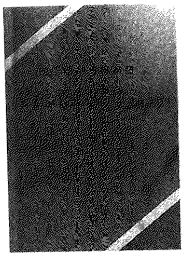
製品名：Let's C+CSD
価格：25,000円
問い合わせ先：パーソナルメディア㈱パッケージソフトウェア営業部
03-3490-8841

Visual C

BASICライクな操作環境をもつCインタプリタ.load, save, files, run, list, editなどのコマンドを使って操作する.作成したソースプログラムはLattice C Ver.3.1以降,Microsoft C Ver.3.0以降の各コンパイラとほぼ互換性がある.NEC PC-9800シリーズ(LT/XAおよびハイレゾモードでの使用を除く)で動作する.


Visual C


ORCHID C



製品名：C統合化システム Visual C ver.J1.01
価格：12,800円
問い合わせ先：㈲データワン
0472-41-5641

ORCHID C

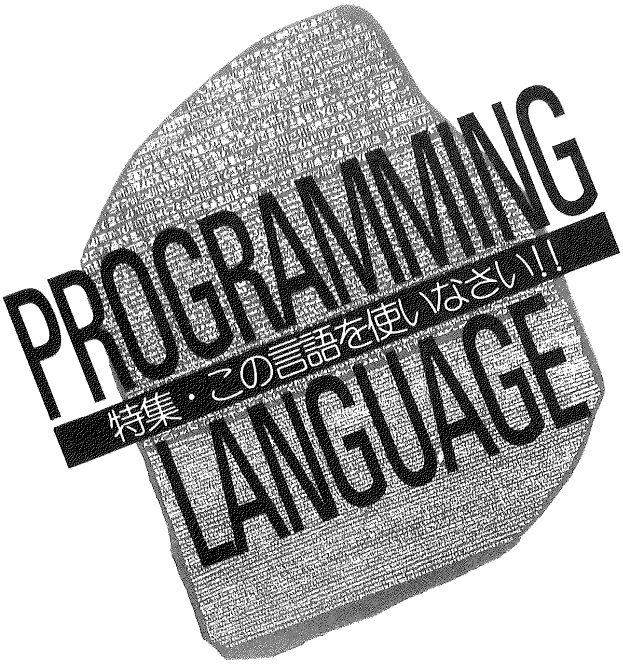
NEC PC-9800シリーズ用に開発された学習・教育用Cインタプリタ.各種のコマンドとファンクションキー,ポップアップメニューを使ってプログラムを作成,実行することができる.コマンド操作でプログラムを参照,実行するコマンドモードと,プログラムの内容を編集するエディットモードがある.

プログラムのトレースやシングルステップ実行,ブレークポイントやウォッチ変数の設定などデバッグ機能も充実しており,一見統合環境コンパイラ風.関数の使用例を示す豊富なサンプルプログラムが付属し,マニュアルでも関数ごとに例題をあげて解説するなどC言語の入門者向き.

製品名：ORCHID C
価格：20,000円
問い合わせ先：オーキット
03-3827-7496

処理系紹介

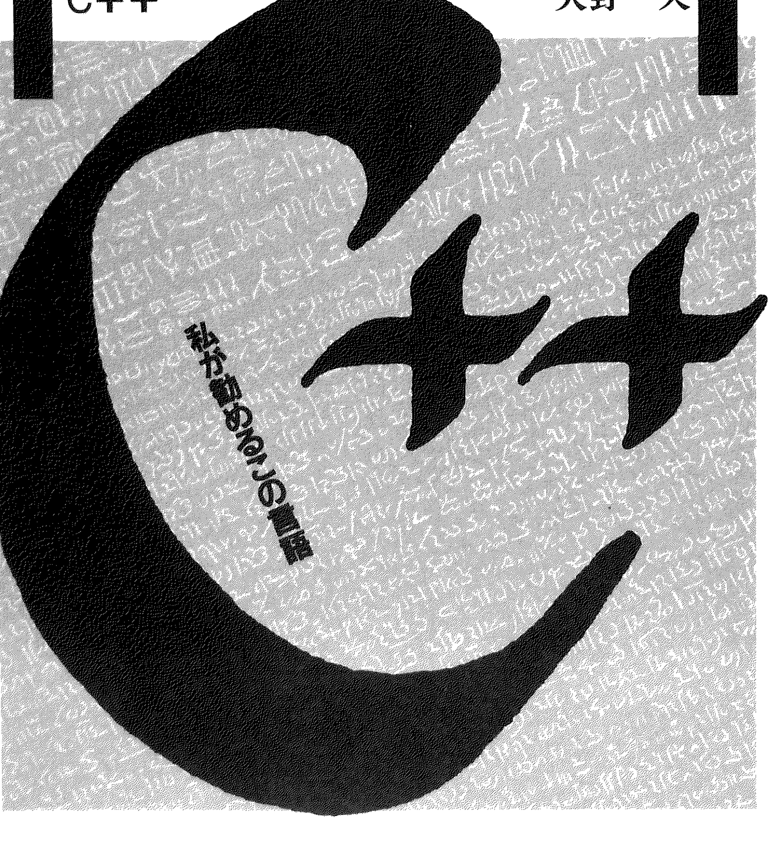
プログラミング用語
構造体　必ずしも同一の型ではない複数のメンバーが集合したデータ構造



オブジェクト指向を取り入れた 次世代C言語

C++

大野一久



C++の魅力

C++がなぜ好きかという質問に簡潔に答えるのは，むずかしい．Cでもかまわないといえばかまわないのだが，Cで，こういうことができたらいいなぁと思っていたことができるようになっているので，やはりC++はいいのである．

C++は，基本的にCのスーパーセットなので(このため，初期のC++の製品はC++のソースからCのソースを吐き出すトランスレータばかりで，ネイティブコードを出力するコンパイラはなかった．今ではコンパイラも発売されている)，Cのもつ便利さのすべてを引きついでいる．このことはまた，CからC++へ移行してもそれまでに築き上げてきたCのソフトウェア資産が無駄にならないということも意味している．この点は，プロのプログラマにとっては重要なポイントだろう．

C++の最大の魅力は，クラスの概念が取り入れられたことによってオブジェクト指向プログラミングが可能になった点である．ページ数の関係で，オブジェクト指向の魅力についての説明は省略するが，現場のプログラマにとってじつに便利な技法（?）だということだけは述べておきたい．

Simula+C =C++?

C++が生まれたのはアメリカのベル研究所であり，ここはCの生まれ故郷でもある．C++の作者であるBjarne Stroustrupは，デンマークに生まれ，イギリスのケンブリッジ大学で博士号を取得したヨーロッパ人だ．彼は

プログラミング用語
再帰呼び出し　自分自身を呼び出す関数呼び出しである．

大学在学中に，ノルウェーで生まれたSimulaというプログラミング言語を利用している．このシミュレーションを行うために開発されたSimulaこそ，すべてのオブジェクト指向言語の源流として知られる言語である．アラン・ケイもSmalltalkを開発する際にSimulaを参考にしている．

このような経歴をもつBjarne StroustrupがCの故郷であるベル研究所にやってきたのだから，彼がCにSimulaの機能を取り入れてC++を作ったのも当然といえるだろう．

実際には，1980年から開発が始められ，最初はSimulaのもつクラスや導出クラスの概念を取り入れた〝クラスつきC（C with Classes）〟として発表された．C++と呼ばれるようになったのは1983年のことで，1985年には最初の製品が発売になっている．

Bjarne Stroustrupは現在もC++の改良を続けており，今発売されている製品の多くがリリース2.xに準拠したものとなっている．

最近では，C++のANSI標準化といった動きもみられるようになり，またMicrosoft社が自社のCコンパイラを将来C++化すると発表するなど，C++がCの後継言語であるということが社会的にも認められ始めたようだ．

クラスとメンバ関数

C++の言語仕様のなかでもっとも重要なのは，クラスとメンバ関数だろう．簡単にいうと，クラスはC言語にある構造体を拡張したもので，プログラマは新しい型としてクラスを宣言する．クラスは構造体同様メンバをもつことができるが，構造体と違って関数をメンバとしてもつことができるようになっている．このような関数をメンバ関数と呼ぶ．

ひとつのクラスは公開部分と未公開部分からなり，この未公開部分にデータメンバを記述することでクラスの外側から直接データをアクセスすることを禁止することが可能となる．データへのアクセスは，公開部分に書かれたメンバ関数にアクセスすることで行われるわけである．こうすることで，オブジェクトのブラックボックス化が可能になり，結果的にプログラム全体の見通しがよくなり，デバッグやコーディングの効率がアップすることになる．

クラスとメンバ関数以外にも，多重継承をはじめとして数多くの機能がC++に取り入れられているが，これらについては実際にC++のプログラミングを行ってそのよさを実感してもらいたい．

サンプルリスト

```
#include <stdio.h>

class device {
    int page;
public:
    virtual void dprintf(char *fmt, ...) = 0;
    virtual char *dname(void) = 0;
    device() { page = 1; }
    device& operator++ (void) {
        dprintf("¥npage %d (%s)¥n", page++, dname());
        return *this;
    }
    device& operator<< (char c) { dprintf("%c", c); return *this; }
    device& operator<< (int n) { dprintf("%d", n); return *this; }
    device& operator<< (char *s) { dprintf("%s", s); return *this; }
};

#define VARG(fmt) ((char *)&(fmt) + sizeof (fmt))

class sout : public device {
protected:
    void dprintf(char *fmt, ...) { vprintf(fmt, VARG(fmt)); }
    char *dname(void) { return "standard output"; }
};

class serr : public device {
protected:
    void dprintf(char *fmt, ...) { vfprintf(stderr, fmt, VARG(fmt)); }
    char *dname(void) { return "standard error output"; }
};

class sprn : public device {
protected:
    void dprintf(char *fmt, ...) { vfprintf(stdprn, fmt, VARG(fmt)); }
    char *dname(void) { return "standard printer"; }
};

main()
{
    sout out;
    out << "string " << 1234 << '¥n';
    ++out;
    out << "This is a test.¥n";
    ++out;
    sprn prn;
    prn << "This is a test.¥n";
    ++prn;
    return 0;
}
```

本物のプログラマのための実践言語

C++がその威力を発揮するのは，大規模なアプリケーション開発に使用されたときだろう．オブジェクト指向を導入したことで，プログラムのカプセル化，部品化が可能になり，またプログラマが解決しようとしている実際の問題とソースコードとが一致するようになるため，プログラムを容易に理解できるようになるからである．

このような高レベルの機能をもちながらも，C++はC言語の高級アセンブラとしての機能もしっかりと継承している．このため，C同様キレのいいプログラミングが可能となる．

C++は，学者が机上で作り出した〝理想のプログラミング言語〟ではない，プログラマがプログラミングの現場から作り出した〝プログラマのための実践的プログラミング言語〟なのである．もしもあなたが本物の，あるいは本物を目指しているプログラマならば，迷わずC++に取り組んで欲しい．けっして損はしないはずである．

最後に簡単なC++のリストをあげておく．このような短いソースで，C++の特徴をすべて示すというのはどだい無理な話なのだが，C++の雰囲気だけでも味わっていただければ幸いである．

プログラミング用語
最適化　高級言語において無駄なコードを省き，プログラムサイズを小さく，または実行速度を速くするためにプログラムを加工すること．

処理系紹介

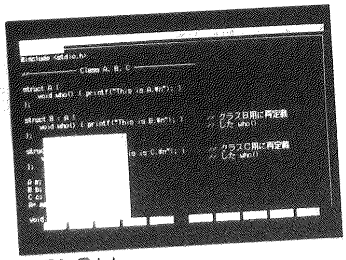
C++

Turbo C++

統合環境と高速コンパイル，強力なデバッグ機能が特長の米Borland社のCコンパイラTurbo Cにオブジェクト指向プログラミングを取り入れた製品．AT&T C++ Ver.2.0に準拠．C++のネイティブコンパイラでありながらコンパイル時のオプションによってANSI C対応のCコンパイラとしても使用できるという欲張った仕様になっている．

製品構成としては標準のTurbo C++，TurboC++にTurbo DebuggerとTurbo Assemblerをバンドルした Turbo C++ Professionalがある．NEC PC-9800シリーズ，EPSON PC-286シリーズ（ハイレゾモード対応）上で動作する．

製品名：Turbo C++ 1.0
価格：29,800円（'91年3月末まで）
製品名：Turbo C++ Professional
価格：49,800円（'91年3月末まで）
問い合わせ先：㈱ボーランドジャパン
03-3486-1400

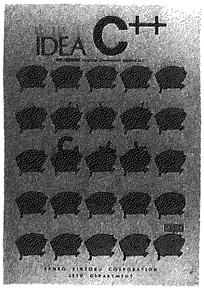


IDEA C++

IDEA C++

中間言語を使用するC++インタプリタ．AT&T C++ Ver.1.2に準拠（91年にVer.2.0対応版発売予定）．MIFESライクなフルスクリーンエディタとソースレベルデバッガを内蔵．統合環境からソースプログラムの中間言語へのコンパイル，リンク，中間言語インタプリタの実行，デバッグ等が行える．MIWA C++トランスレータ，Microsoft Cコンパイラに対応．IDEA C++で作成したソースプログラムをこれらのソフトウェアを用いてEXEファイルに変換することができる．また，ユーザーが作成した関数をIDEA C++に組み込み可能なIDEA C++ EXもある．

NEC PC-9800シリーズ(XA，LTを除くノーマルモードのみ)，EPSON PC-286シリーズで動作．



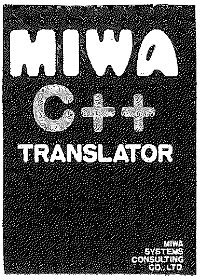
製品名：IDEA C++
価格：29,800円
問い合わせ先：三光金属㈱シート事業部
0729-65-7225

MIWA C++

日本語対応のC++トランスレータ．Cの上位言語としてC++を位置づけ，C++で記述したソースプログラムを対応するCコンパイラ用のCソースプログラムに変換する．したがって，実行形を作成するには別途Cコンパイラを用意する必要がある．

対応するCコンパイラのすべてのメモリモデルをサポート．MS-DOSシステムコールの使用が可能．コンパイルとリンクをスムーズに行うためのコンパイルドライバが付属．さらにサンプルプログラムも添付されている．

NEC PC-9800シリーズ，EPSON PC-286シリーズ，富士通FMR-60，IBM5560，東芝J-3100シリーズ等で動作確認済み．



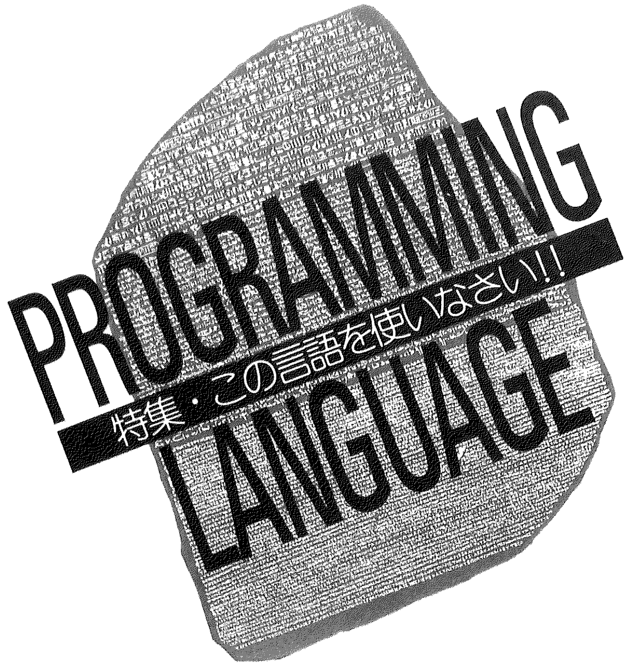
製品名：MIWA C++ Ver.1.2（Microsoft C Ver.4.0，5.1対応版）
価格：59,800円
製品名：MIWA C++ Ver.1.2（TURBO C Ver.1.5，2.0 対応版）
価格：59,800円
問い合わせ先：㈱ミワシステムスコンサルティング
045-312-5418

Zortech C++

英Zortech社のC++コンパイラ．MS-WINDOWS 3.0にも対応した286用と，DOS Extenderを含む386用がある．

製品名：Zortech C++ V/2.1 開発用パッケージ（MS-DOS版/286用）
価格：168,000円
製品名：Zortech C++/386 V/2.1開発用パッケージ
価格：280,000円
問い合わせ先：マイクロソフトウェア㈱
03-3813-8221

プログラミング用語
自動変数　プログラムの実行時にメモリに確保される変数のこと．



大型機の経験を生かせる互換性

COBOL　海老原浩之

COBOL(コボル)とは,「Common Business Oriented Language」が正式の呼び方で，各単語の頭文字をとって略したいい方なのです．日本語に訳すと「事務処理共通言語」といった意味になります．

なぜ,COBOLが好きなのか?

なんといっても，大型計算機やミニコン,オフコンで培われてきた技術が,そのまま生かせることですね．COBOLといえば，大型計算機やミニコン，オフコンではあたりまえに使われている言語ですが，パソコンではあまり使われない言語です．しかし，大型機やミニコン，オフコンといった環境でCOBOLをあたりまえに使っていたプログラマにとって，パソコン上で事務処理アプリケーションを開発するにはいちばん使いやすい言語であり，開発効率が高いのです．

COBOLは，言語そのものの互換性が非常に高いので，大型計算機やミニコン，オフコンで組んだプログラムをパソコンに移植することが非常に短期間でできます．また，パソコンの代表言語としてBASICがありますが，BASICで事務処理プログラムを作ると，非常に手間と労力がいり，時間もかかります．しかし，COBOLは事務処理を目的として開発された言語なので,時間や手間がBASICの半分ですむのです．

COBOLの生い立ち

COBOLの歴史は,1959年にアメリカの国防省が事務データ処理のための共通言語の必要性を提唱し,民間企業や政府

プログラミング用語
シフトJIS　JIS漢字コードをシフトし，グラフィック文字部に割り当てたコード体系．MS-DOSや一部のUNIXで採用されている．

諸機関，さらにはコンピュータメーカーがそれに応じたことにはじまります．

そして，CODASYL (Conference On Data System Language) が発足し，CODASYLとほかに３つの委員会が実際の言語仕様を規定し，CODASYLが1960年に第１回目のCOBOL言語の仕様を発行しました．これがCOBOL-60と呼ばれる最初のCOBOL言語仕様です（COBOLでは，規定された仕様を「COBOL-xx」といったふうに呼びます．xxの所に仕様が規定された年が入り，現在はCOBOL-85の仕様が使われています）．

またCODASYLでは定期的に，COBOLの言語仕様の改訂を行い，この作業はつねに継続して行われています．そのようにして，データ処理分野におけるさまざまな要求や環境の変化に応じてCOBOLの仕様も変化を遂げてきました．

CODASYLが規定したCOBOLの言語仕様を元にして，アメリカ標準規格（ANSI）→国際標準規格（ISO）→日本工業規格（JIS）というようにしてCOBOLの規格が設定されます．このような規格化では，言語仕様の規定から数年遅れるので，規格そのものは必ずしも最新の時代要請を反映してはいないといえます．しかし，コンピュータメーカーが作成するCOBOLは，規格化に先駆けてメーカーが独自に新しい機能を取り入れています．したがって，ある意味では市販されているCOBOLは，規格されたものよりも機能が充実しているので，より使いやすい言語になっているといえます．

言語仕様上の特徴

COBOLの特徴をあげると，「言語の互換性」「理解のしやすさ」「事務処理プログラム開発に適している」の３点といえます．

① 言語の互換性に関しては，COBOLの命令が規格化によって標準化されているため，どのようなコンピュータ上のCOBOLでも，メーカー独自の言語仕様の違いさえ把握すればプログラムの作成を行うことができます．またCOBOLの機能や形式をCODASYLが規定し，さらにANSI, ISO, JISなどでしっかりと規格化するため，COBOL本来の機能に関しては互換性が保証されます．

② 理解のしやすさに関しては，プログラムそのものが簡単な英文形式で，文と数式で書くのが基本となるため，言語を理解するのが早いといえます．

③ 事務処理プログラム開発に適しているという点に関しては，COBOLの誕生の背景が事務処理に適した言語の開発を基本コンセプトにしているのですからいうまでもありません．報告書作成機能や多数のファイルの管理など，事務処理に必要な機能が充実しています．このため，帳票などは簡単に作成でき，ファイルの扱いも，「索引編成」「順編成」「相対編成」など用途に応じて使いわけができるので，ファイルの扱いに悩されないですみます．

最後に，NECより発売されているPC-COBOLで書いた住所管理プログラムの例を示しておきます．

サンプルリスト

```
000010 IDENTIFICATION  DIVISION
000020*                個人情報管理プログラム
000030*AUTHOR.         HIROYUKI EBIHARA
000040*COMPILER        NEC PC-COBOL
000050 PROGRAM-ID .    ADDCBL   .
000060 ENVIRONMENT     DIVISION .
000070 CONFIGURATION   SECTION .
000080 SOURCE-COMPUTER . PC-9801 .
000090 OBJECT-COMPUTER . PC-9801 .
000100 SPECIAL-NAMES . GDD IS GUIDE .
000110*****************************************************************
000120*    ファイルの管理方法の定義
000130*****************************************************************
000140 INPUT-OUTPUT    SECTION .
000150 FILE-CONTROL .
000160      SELECT ADDFILE ASSIGN TO MSD
000170              ORGANIZATION   INDEXED
000180              ACCESS MODE IS DYNAMIC
000190              RECORD KEY IS  ADD-KEY
000200              STATUS FILE-STS .
000210*
000220 DATA            DIVISION .
000230*****************************************************************
000240*    ファイルのレコードの定義
000250*****************************************************************
000260 FILE            SECTION .
000270 FD  ADDFILE LABEL RECORD STANDARD
000280     BLOCK CONTAINS  1    RECORDS
000290     VALUE OF IDENTIFICATION "ADDRESS.DAT" .
000300 01  ADD-REC .
000310     02     ADD-KEY .
000320        03  ADD001    PIC 9(004) .                          整理番号
000330     02     ADD002    PIC N(008) .                          氏名
000340     02     ADD003    PIC X(006) .                          郵便番号
000350     02     ADD004    PIC X(030) .                          住所1
000360     02     ADD005    PIC X(030) .                              2
000370     02     ADD006    PIC X(030) .                              3
000380     02     ADD007    PIC X(013) .                          電話番号
000390     02     ADD008    PIC X(013) .                          FAX 番号
000400     02     ADD009          .                               誕生日
000410          03 ADD009A  PIC 9(002) .                          年
000420          03 ADD009B  PIC 9(002) .                          月
000430          03 ADD009C  PIC 9(002) .                          日
000440     02     ADD0010   PIC N(001) .                          血液型
000450     02     ADDA011  PIC X(030) .                           備考1
000460     02     ADDA012  PIC X(030) .                           備考2
000470     02     ADDA013  PIC X(030) .                           備考3
000480*    02     FILLER   PIC X(016) .
000490*****************************************************************
000500*    作業用変数等の定義
000510*****************************************************************
000520 WORKING-STORAGE  SECTION .
000530 77  FILE-STS PIC XX .
000540 77  F10-K     PIC XX   VALUE  "10" .
000550 77  WRT-K     PIC XX   VALUE  "07" .
000560 77  MSG01     PIC X(8) VALUE  "KB ｴﾗｰ  " .
000570 77  MSG02     PIC X(8) VALUE  "ｶｷｺﾐ ｴﾗｰ" .
000570 77  MSG03     PIC X(8) VALUE  "ｻｸｼﾞｮｴﾗｰ" .
000580 77  SEL       PIC 9    VALUE  ZERO  .
000590 77  KAKU      PIC 9    VALUE  ZERO  .
000600 01  FILE-ERR-MSG .
000610     02  FMSG01      PIC X(9) VALUE  "FILE ｴﾗｰ:" .
000620     02  FMSG02      PIC X(2) VALUE  SPACE     .
000630 01  IN-REC .
000640     02     SCRA001  PIC 9(004) .
000650     02     SCRA002  PIC N(008) .
000660     02     SCRA003  PIC X(006) .
000670     02     SCRA004  PIC X(030) .
000680     02     SCRA005  PIC X(030) .
000690     02     SCRA006  PIC X(030) .
000700     02     SCRA007  PIC X(013) .
000710     02     SCRA008  PIC X(013) .
000720     02     SCRA009 .
000730            03 SCRA009A PIC 9(002) .
000740            03 SCRA009B PIC 9(002) .
000750            03 SCRA009C PIC 9(002) .
000760     02     SCRA010  PIC N(001) .
000770     02     SCRA011  PIC X(030) .
000780     02     SCRA012  PIC X(030) .
000790     02     SCRA013  PIC X(030) .
000800*****************************************************************
000810*    表示画面、データ入力部の定義
000820*****************************************************************
000830 SCREEN  SECTION .
000840 SD  GAMEN END STATUS IS  ENDSTS
000850 01      SCR-MENU   CLEAR SCREEN .
000860     02 LINE  1 COLUMN  29 PIC N(011)
000870        VALUE NC"個 人 情 報 管 理" COLOR CYAN .
000880     02 LINE  4 COLUMN  31 PIC N(009)
```

```
           VALUE NC"1. 登        録"    COLOR WHITE .
       02 LINE  7 COLUMN  31 PIC N(009)
           VALUE NC"2. 修        正"    COLOR WHITE .
       02 LINE 10 COLUMN  31 PIC N(009)
           VALUE NC"3. 削        除"    COLOR WHITE .
       02 LINE 13 COLUMN  31 PIC N(009)
           VALUE NC"9. 終        了"    COLOR WHITE .
       02 LINE 19 COLUMN  30 PIC N(009)
           VALUE NC"番号を選んで下さい"    COLOR WHITE .
       02 LINE 19 COLUMN  51 PIC X(003)
           VALUE   "[ ]" .
01         SCR-DISP   CLEAR SCREEN .
       02 LINE  1 COLUMN  27 PIC N(011)
           VALUE NC"個 人 情 報 管 理"   COLOR CYAN  .
       02 LINE  3 COLUMN   6 PIC N(003)
           VALUE NC"整理No" .
       02 LINE  3 COLUMN  12 VALUE ":[    ]" .
       02 LINE  3 COLUMN  24 PIC N(002)
           VALUE NC"氏名" .
       02 LINE  3 COLUMN  28 VALUE ":[                    ]" .
       02 LINE  5 COLUMN   6 PIC N(004)
           VALUE NC"郵便番号" .
       02 LINE  6 COLUMN  13 VALUE "1 " .
       02 LINE  7 COLUMN   6 PIC N(003)
           VALUE NC"住  所" .
       02 LINE  7 COLUMN  13 VALUE "2 " .
       02 LINE  8 COLUMN  13 VALUE "3 " .
       02 LINE  9 COLUMN   6 PIC N(004)
           VALUE NC"電話番号" .
       02 LINE 10 COLUMN   6 VALUE "FAX" .
       02 LINE 10 COLUMN  10 PIC N(002)
           VALUE NC"番号" .
       02 LINE 11 COLUMN   6 PIC N(003)
           VALUE NC"誕生日" .
       02 LINE 11 COLUMN  16 VALUE "  /  /" .
       02 LINE 12 COLUMN   6 PIC N(004)
           VALUE NC"血液型  " .
       02 LINE 12 COLUMN  21 PIC N(001)
           VALUE NC"型" .
       02 LINE 14 COLUMN   6 PIC N(004)
           VALUE NC"備    考" .
       02 LINE 22 COLUMN  20 PIC N(002)
           VALUE NC"確認" .
       02 LINE 22 COLUMN  25 VALUE "[ ]  Y:" .
       02 LINE 22 COLUMN  32 PIC N(002)
           VALUE NC"実行" .
       02 LINE 22 COLUMN  38 VALUE "N:" .
       02 LINE 22 COLUMN  40 PIC N(002)
           VALUE NC"取消" .
       02 LINE  5 COLUMN   5
               OVER LINE TO      22
               VERTICAL LINE AT      5  15  22                  .
       02 LINE  6 COLUMN   5
               OVER LINE TO      65
               VERTICAL LINE AT      5  15  65
               OCCURS              7                            .
       02 LINE 12 COLUMN   5
               OVER LINE TO      65 UNDER LINE TO     65
               VERTICAL LINE AT      5  15  65                  .
       02 LINE 14 COLUMN   5
               OVER LINE TO      15
               VERTICAL LINE AT      5  15                      .
       02 LINE 15 COLUMN   5
               VERTICAL LINE AT      5  15                      .
       02 LINE 16 COLUMN   5
               UNDER LINE TO     15
               VERTICAL LINE AT      5  15                      .
       02 LINE 14 COLUMN  15
               OVER LINE TO      65
               VERTICAL LINE AT      15  65
               OCCURS              2                            .
       02 LINE 16 COLUMN  15
               OVER LINE TO      65 UNDER LINE TO     65
               VERTICAL LINE AT      15  65                     .
01         USE-DATA-MSG .
       02 LINE 24 COLUMN 015 PIC N(014)
           VALUE NC"既にデータが登録されています" .
01         NO-DATA-MSG .
       02 LINE 24 COLUMN 015 PIC N(014)
           VALUE NC"指定されたデータはありません" .
01         CLEAR-MSG .
       02 LINE 24 COLUMN 015 PIC X(028)
           VALUE  "                            " .
01         SCR-KEY .
       02    SCRG001
           LINE 03 COLUMN 014 PIC 9(004)  USING     SCRA001
             CHECK OVERFLOW                                       .
01         SCR-IN .
       02    SCRG002
           LINE 03 COLUMN 030 PIC N(008)  USING     SCRA002
             CHECK OVERFLOW                                       .
       02    SCRG003
           LINE 05 COLUMN 016 PIC X(006)  USING     SCRA003
             CHECK OVERFLOW                                       .
       02    SCRG004
           LINE 06 COLUMN 016 PIC X(030)  USING     SCRA004
             CHECK OVERFLOW                                       .
       02    SCRG005
           LINE 07 COLUMN 016 PIC X(030)  USING     SCRA005
             CHECK OVERFLOW                                       .
       02    SCRG006
           LINE 08 COLUMN 016 PIC X(030)  USING     SCRA006
             CHECK OVERFLOW                                       .
       02    SCRG007
           LINE 09 COLUMN 016 PIC X(013)  USING     SCRA007
             CHECK OVERFLOW                                       .
       02    SCRG008
           LINE 10 COLUMN 016 PIC X(013)  USING     SCRA008
             CHECK OVERFLOW                                       .
       02    SCRG009  LINE 11 .
           03      COLUMN 016 PIC X(002)  USING     SCRA009A      .
           03      COLUMN 019 PIC X(002)  USING     SCRA009B      .
           03      COLUMN 022 PIC X(002)  USING     SCRA009C
                                 CHECK OVERFLOW                   .
       02    SCRG010
           LINE 12 COLUMN 017 PIC N(001)  USING     SCRA010
             CHECK OVERFLOW                                       .
       02    SCRG011
           LINE 14 COLUMN 016 PIC X(030)  USING     SCRA011
             CHECK OVERFLOW                                       .
       02    SCRG012
           LINE 15 COLUMN 016 PIC X(030)  USING     SCRA012
             CHECK OVERFLOW                                       .
       02    SCRG013
           LINE 16 COLUMN 016 PIC X(030)  USING     SCRA013
             CHECK OVERFLOW                                       .
01   SENTAKU .
       02 LINE 19 COLUMN 052 PIC X(001)  INTO      SEL
             CHECK OVERFLOW                                       .
01   KAKUNIN .
       02 LINE 22 COLUMN 026 PIC X(001)  INTO      KAKU
             CHECK OVERFLOW                                       .
*
 PROCEDURE       DIVISION .
*************************************************************************
*    プログラムの開始
*************************************************************************
 PROG-BGN .
       OPEN I-O    ADDFILE  .
       IF FILE-STS = ZERO  GO TO SELECT-RTN .
       IF FILE-STS = 91    CLOSE ADDFILE
                           OPEN  OUTPUT ADDFILE
                           CLOSE ADDFILE  GO  TO  PROG-BGN .
       MOVE  FILE-STS  TO  FMSG02 .
       DISPLAY FILE-ERR-MSG  UPON  GUIDE .
       GO TO END-RTN  .
*************************************************************************
*    処理の選択
*************************************************************************
 SELECT-RTN .
       DISPLAY      SCR-MENU .
 SEL-01  .
       ACCEPT       SENTAKU  .
 SEL-02  .
       IF SEL =  1  GO   TO   ADD-RTN   .                   登録
       IF SEL =  2  GO   TO   UPD-RTN   .                   修正
       IF SEL =  3  GO   TO   DEL-RTN   .                   削除
       IF SEL =  9  GO   TO   END-RTN   .                   終了
       GO      TO     SEL-01  .
*************************************************************************
*    データの登録処理
*************************************************************************
 ADD-RTN .
       MOVE  SPACE  TO IN-REC .
       DISPLAY    SCR-DISP  .
 ADD-01 .
       ACCEPT     SCR-KEY    .
       IF ENDSTS  = F10-K   GO   TO   SELECT-RTN .
       MOVE  SCRA001  TO  ADD001 .
       READ  ADDFILE  INVALID  DISPLAY CLEAR-MSG
                               GO   TO  ADD-02 .
       DISPLAY    USE-DATA-MSG   .
       GO    TO   ADD-01 .
 ADD-02 .
       ACCEPT     SCR-IN      .
       IF ENDSTS  = F10-K   GO   TO   SELECT-RTN .
       IF ENDSTS  = WRT-K   GO   TO   KAKU-RTN .
       GO    TO   KAKU-RTN .
*************************************************************************
*    データの修正処理
*************************************************************************
 UPD-RTN .
       MOVE  SPACE  TO IN-REC .
       DISPLAY    SCR-DISP    .
 UPD-01 .
       ACCEPT     SCR-KEY    .
       IF ENDSTS  = F10-K   GO   TO   SELECT-RTN .
       MOVE  SCRA001  TO  ADD001 .
       READ  ADDFILE  INVALID  DISPLAY  NO-DATA-MSG
                               GO   TO  UPD-01 .
       MOVE  ADD-REC  TO  IN-REC .
       DISPLAY    SCR-IN  CLEAR-MSG .
 UPD-02 .
       ACCEPT     SCR-IN      .
       IF ENDSTS  = F10-K   GO   TO   SELECT-RTN .
       IF ENDSTS  = WRT-K   GO   TO   KAKU-RTN .
       GO    TO   KAKU-RTN .
*************************************************************************
*    データの削除処理
*************************************************************************
 DEL-RTN .
       MOVE  SPACE  TO IN-REC .
       DISPLAY    SCR-DISP    .
 DEL-01 .
       ACCEPT     SCR-KEY    .
       IF ENDSTS      = F10-K GO TO SELECT-RTN .
       MOVE  SCRA001  TO  ADD001 .
       READ  ADDFILE  INVALID  DISPLAY  NO-DATA-MSG
                               GO   TO  UPD-01 .
       MOVE  ADD-REC  TO  IN-REC .
       DISPLAY    SCR-IN  CLEAR-MSG .
       GO    TO   KAKU-RTN  .
*************************************************************************
*    確認の作業
*************************************************************************
 KAKU-RTN .
       ACCEPT     KAKUNIN .
       IF KAKU = "Y"  PERFORM WRITE-RTN THRU WRITE-EXT
                      GO   TO   SEL-02 .
       IF KAKU = "N"  GO   TO   SEL-02 .
       GO    TO   KAKU-RTN .
*************************************************************************
*    プログラムの終了
*************************************************************************
 END-RTN .
       CLOSE    ADDFILE .
       STOP     RUN .
*************************************************************************
*    データの書き込み、削除を行うサブルーチン.
*************************************************************************
 WRITE-RTN .
       MOVE   IN-REC   TO   ADD-REC .
       IF SEL = 1    GO  TO  WRITE-01  .
       IF SEL = 2    GO  TO  WRITE-02  .
       IF SEL = 3    GO  TO  WRITE-03  .
 WRITE-01  .
       WRITE   ADD-REC INVALID GO  TO  WRITE-02 .
       GO        TO      WRITE-EXT  .
 WRITE-02  .
       REWRITE ADD-REC INVALID DISPLAY MSG02 UPON GUIDE .
       GO        TO      WRITE-EXT  .
 WRITE-03  .
       DELETE  ADDFILE INVALID DISPLAY MSG03 UPON GUIDE .
 WRITE-EXT .
       EXIT  .
```

プログラミング用語
スタック　データの記憶方法のひとつ．データをスタック領域に退避し，要求があった場合最後に退避したデータから取り出す．

処　理　系　紹　介

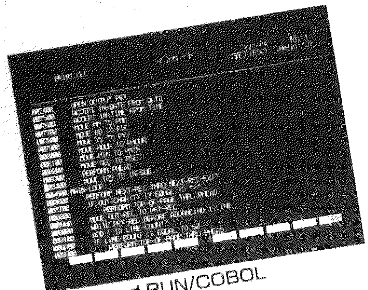
COBOL

Advanced RUN/COBOL

COBOL言語の学習を目的とした世界唯一のCOBOLインタプリタ．プログラミング言語の学習環境を提供することを目的としたライフボート社のAdvanced RUNシリーズのひとつ．ANSI-1974第1水準に準拠する．

インターフェースはBASICライクなコマンドベース，専用エディタとソースレベルデバッガを内蔵．NEC PC-9801シリーズ，富士通FMR50/60/70，FM TOWNS，東芝J-3100シリーズ，IBM PS/55シリーズに対応．

製品名：Advanced RUN/COBOL
価格：29,800円
問い合わせ先：(株) ライフボート
03-3293-4711


Advanced RUN/COBOL

日本語Master COBOL-74

パソコン，UNIXマシン用COBOLコンパイラ．ANSI-1974レベルII完全準拠．機種別のランタイムシステム（RTS）のもとで，同一のオブジェクトプログラムの実行が可能．また画面の定義や日本語処理等の各種拡張機能を備える．C言語やこれと同じ関数呼出し規約をもつ各種言語とのリンクが可能．

MS-DOS版はNEC PC-9800/N5200シリーズ，富士通FMRシリーズ，東芝J-3100/J-3300シリーズ，松下Panacomシリーズ，各社AXシリーズ等各社のパソコンに対応．

製品名：日本語Master COBOL-74(MS-DOS版)
価格：150,000円
問い合わせ先：(株) コミュニケーションサイエンス
03-3793-6751

日本語LEVEL II COBOLコンパイラ

英Micro Focus社のCOBOLコンパイラ．NEC PC-9800シリーズ，PC-H98シリーズで動作．

製品名：日本語LEVEL II COBOLコンパイラ
価格：195,000円または196,000円（メディアによる）
問い合わせ先：日本電気（株）パソコンインフォメーションセンター
03-3452-8000

日本語COBOL/2コンパイラ

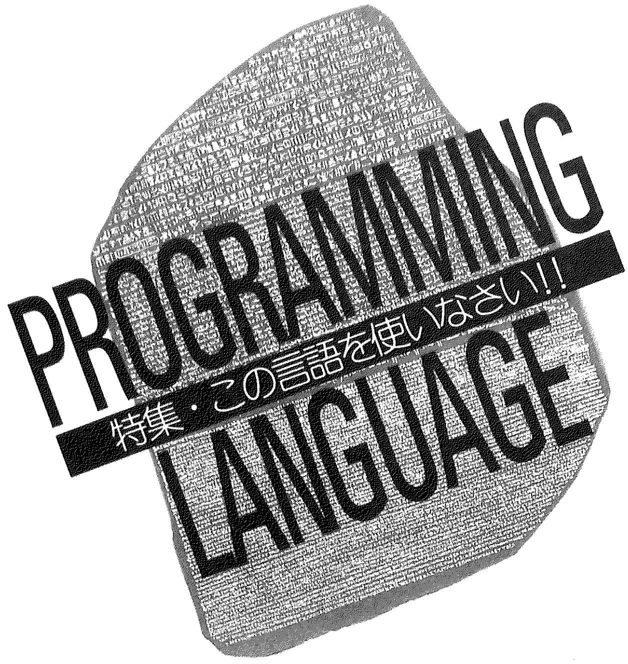
NEC PC-9800シリーズ，PC-H98シリーズ用COBOLコンパイラ．MS-DOS上，OS/2上のいずれでも利用可能．日本語LEVEL II COBOLとソースレベルで互換性がある．

製品名：日本語COBOL/2コンパイラ Ver.1.1A
価格：195,000円または196,000円（メディアによる）
問い合わせ先：日本電気（株）パソコンインフォメーションセンター
03-3452-8000

日本語PC-COBOL

NEC PC-9800シリーズ，PC-H98シリーズ用COBOLコンパイラ．

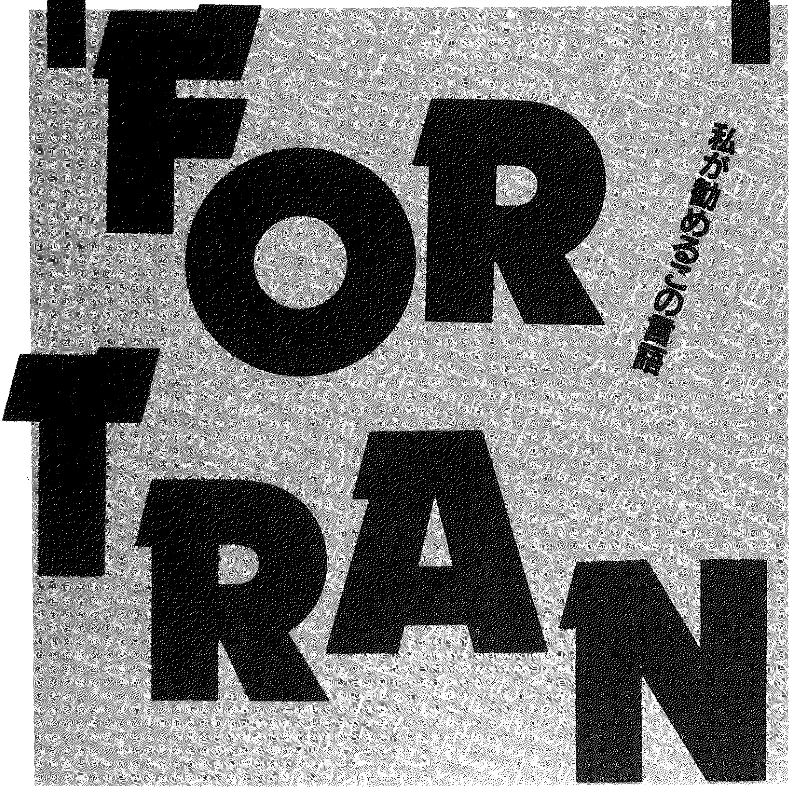
製品名：日本語PC-COBOLコンパイラ Ver1.1（ノーマルモード対応）
価格：99,000円または100,000円（メディアによる）
製品名：日本語PC-COBOLコンパイラ Ver.1.1A（ハイレゾモード対応）
価格：100,000円
問い合わせ先：日本電気（株）パソコンインフォメーションセンター
03-3452-8000



科学技術計算の世界共通語

FORTRAN

金属材料技術研究所 桜井健次



数値計算に莫大なライブラリを蓄積

みなさんは外国語を学ぼうとするとき，どこの国の言葉を選ばれるでしょうか？　もちろん，どの国の言葉もその国の文化や人々の暮らしを知るうえで重要ですし，とくにそれぞれの間になにか優劣があるわけではありません．にもかかわらず，そのなかで英語やフランス語が，かなり多くの人々によって学ばれているのはなぜでしょうか？ それはさまざまな歴史的な経緯のなかで，結果的に世界共通言語としての性格を帯びているからです．多くの重要なことがらが，これらの言語により書かれ，また話されています．もし，私たちがその内容を知り，また自らも関わってゆこうと思うならば，その言語を学ぶことは必須となりましょう．

プログラミング言語についても似たようなことがいえます．FORTRANは世界最初の実用的なプログラミング言語として生まれ，その後今日に至るまで，全世界で使われ，無数の資産が築かれてきました．とくに，科学技術計算の分野では，FORTRANの世界共通言語としての地位は絶対的なものです．たしかに，最近はBASICやC，PASCALのための汎用の数値計算ライブラリも種々のマシン用に作成され，市販もされています．しかし，その内容を見れば明らかなように，行列計算や微積分，常微分方程式など，ごく一般的なものに留まっていますし，その解法の選択枝もかなり限られています．また，その絶対数もそれほど多いとはいえません．これに対して，FORTRANには，数値計算でおよそ考えられるもののほとんどすべてをカ

プログラミング用語
制御文　プログラムの実行の流れを変える文で反復文や選択文がある．

バーするライブラリが，それも数え切れないくらい多くの種類あるのです．また，計算機科学はもちろん，素粒子物理学や流体力学，量子化学などの分野におけるプログラムの蓄積も相当なものです．そのようなわけで，科学技術計算の分野で何らかの仕事をするためには，FORTRANをマスターすることは非常に重要です．

資産の公開・共有によって普及

FORTRANという名称は，formula translation (数式翻訳) に由来するもので，1954年にIBM社により数値計算用のプログラミング言語として開発されました．その後，世界中で広く使われるようになるのですが，その過程できちんとした規格がつくられていったおかげで，あるユーザーにより開発されたプログラムが別な計算機センターでも変更なしに動作させるといったことが可能になり，資産の公開・共有が促進され，ますます普及することになりました．

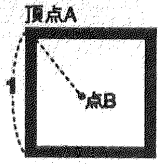
1966年にアメリカでFORTRAN IV (FORTRAN66) が規格化され，日本国内でもこれをもとにJISによる標準化がなされました．さらに，1977年のいわゆるFORTRAN77 (日本国内では1982年にJISにとり入れられた) は，これらに対する上位互換性をもちながら，機能がかなり強化されたために，きわめて強力な標準となりました．今日使用されているFORTRANコンパイラのほとんどが，このFORTRAN77に準拠していることはいうまでもありません．1990年代に入り，新しい規格の導入が日程にのぼっています．これは，互換性を保ちながら，現代化をすすめ，新しい時代の要請に応えてゆこうというものです．

パソコンでの開発が普及する中で

FORTRANはその歴史的経緯からいって，パソコンが普及する以前に，そのプログラミング言語としての地位を確立しました．ですから，もともと大型計算機で使われていたものが，ミニコン・ワークステーションやパソコンの環境でも使われるようになってきたということになります．FORTRANに類似した初心者教育用の言語として開発されたBASICは，その後，対話型のインタプリタ言語としての特徴がパソコンのような環境に適していることから，その標準的な言語となっていったわけですが，このあたりの対比はたいへん興味深いものがあります．私見ですが，FORTRANは大型計算機と切り離してはなかなか考えにくい言語だと思います．ミニコン・ワ

乱数を使って円周率を計算するショートプログラム

1辺の長さが1の正方形の頂点の1つをA，正方形内部の任意の点をBとします．Bをまったくランダムにとったときに AB の長さが1以下になる確率はどれくらいでしょうか？　このプログラムではそれを乱数によってシミュレートします．他方，理論的にはその確率は大数の法則により面積比に漸近し，円周率の4分の1になります．これを使って円周率を求めてみましょう(例：2000回)．試行回数を増やすとそれだけ正確な値に近づくはずですが，一方，乱数の一様性の不十分さも顕在化してきますから，いろいろなシミュレーションに応用するためには乱数を発生する種もいろいろ変えて試してみる必要があります．J-3100(日本語モード) で，MS-FORTRAN 4.1 (IBM PC版) によりコンパイル実行して動作を確認しました．他機種でもたぶん大丈夫でしょう．なお，コンパイル時にwarningが出ても気にする必要はありません．(桜井健次)



●1辺の長さが1の正方形

```
      REAL      X,Y,Z,PI
      INTEGER*4 I,IO,K,N
      WRITE(6,100)
C     -- N は 試行回数 ---
      READ(5,*) N
C     -- K は 試行の結果，条件を満たした回数 ---
      K=0
C     -- IO は乱数発生の種．いろいろ変えてみるとよい ---
      IO=1
C
      DO 10 I=1,N
      CALL RND(IO, X)
      Z=X*X
      CALL RND(IO, Y)
      Z=Z+Y*Y
      IF (Z.LT.1.0) K=K+1
      PI=4.0*FLOAT(K)/FLOAT(N)
   10 CONTINUE
C
      WRITE(6,101) N,K
      WRITE(6,102) PI
C
C     --- FORMAT SENTENCES ------------------------------
  100 FORMAT(' 試行回数は？ ')
  101 FORMAT(' 試行回数                = ', I8,/,
     +       ' ＡＢが１以下だった回数 = ', I8)
  102 FORMAT(' 円周率                  = ', F8.5)
C
      STOP
      END
C
C     ---------------------------------------------------
C     乗積合同法による乱数の発生
C     ---------------------------------------------------
      SUBROUTINE RND(I, X)
      REAL      X
      INTEGER*4  I, M, K
C     -- M は 4 byte 整数型での最大値 (2 の 31 乗から 1 を引く) ---
C     -- M はこのとき素数になっている                           ---
      M=2147483647
C     -- K は M が 2^31-1  の場合に採用される乗数 ---
      K=16807
      I=I*K
C     -- 桁あふれ補正 ---
      IF (I.LT.0) I=I+M+1
      X=FLOAT(I)/FLOAT(M)
      RETURN
      END
```

ークステーションでは必ずしもそうでもないのですが，パソコンでは，まだまだ独自の資産は少ないといわなければなりません.むしろ,FORTRAN77準拠という意味で，大型計算機の上で使用できる形で公開・共有されている莫大な資産が活用できるというメリットが現時点では前面に出ているように思います.

ところで，今日では，大型計算機で動作させるプログラムもその開発環境として，ワークステーションやパソコンが使われることはすでに一般的になっています．使いやすいエディタと通信ソフトをパソコン側で用意していれば，ほとんどの作業が，自分の机の上でできてしまい，ユーザーとしてはたいへん好都合なわけです．また，実験室でデータをパソコンに落とし，それを大型計算機に転送して解析を行うようなこともよく行われています．こうして，パソコン上でのソースプログラムのエディット（または処理すべきデータの準備）～大型計算機へのアップロード～コンパイル・実行～計算結果のダウンロードを何度となく繰り返すことで，作業が進められてゆきます.

ひと昔前ならば，計算はどんな簡単なものでも大型計算機でやるというようなことは珍しいことではありませんでした．しかし，最近のパソコンの高機能化，ラップトップパソコンの登場による1人1台化，そしてLANの充実により，計算機の効率的な組み合わせ利用と使い分けが新しい常識になってしまいました．規格によりソースレベルでの高度な互換性が保障されていることの強みがこのような使い方を容易にしたといえるでしょう．まったく当たり前のことですが，条件が整ってさえいれば，パソコンで十分処理できるものは，パソコン用のコンパイラで片づけるのが合理的であり，これからますます幅広く使われるようになることでしょう.

パソコン上での環境整備の問題

パソコンとひとくちにいっても，いろいろな種類の機械があるわけですが，現在，もっとも多く使われているのは，インテル社の8086(80286あるいは80386も含めて）をベースにした16ビット（または32ビット）パソコンでしょう．これらの機種のOSとしてはMS-DOSが圧倒的に多く使用されています．そして，このMS-DOS上で動作するFORTRANコンパイラが何種類か市販されています．ここでは，パソコン上で動作するプログラムを開発しようとするときに留意すべき点を，とくに大型計算機などとの相違点を中心に述べることにしましょう.

第1は，メモリの制約を考慮する必要があることです．昔は，もともと大型計算機でさえも少ないメモリで計算しなければならないケースは多くあり，メモリを節約するようなプログラムを書くのも技術のうちだったのですが，最近はよほど特別にメモリを使うようなプログラムを別にすればそのようなことは考える必要はなくなりました．パソコンでもメモリそのものはコスト的にも全然問題なくなってきているのですが，8086，そしてそれをベースにしているMS-DOSのメモリ空間の大きさがむしろ制約になっています．しかし，今後は，80286以上のCPUを持つ機種で，MS-Windows 3.0のstandard modeやOS/2を利用することにより，この問題は緩和されることでしょう.

第2は，数値演算プロセッサ(8087・80287・80387）の使用です．FORTRANで実際にやる仕事が科学技術計算であることを考えると，これはぜひ装備することが望まれます．最近のコンパイラは，装着されていない場合でも互換性を保つように，8087のコードを出すようなエミュレータを備えているのが普通ですが，実行速度を考えれば，そのメリットは相当なものです.

第3は，パソコンならではのきめ細かいやり取りを行うためのモジュールを充実させることです．画面制御やグラフィックス，パーソナルプロッタへの出力，RS-232Cを介したデータ通信，GPIBによる機器の制御などは，パソコンならぜひ必要な機能です．パソコン用のBASICなどでは標準のコマンドの形で用意しているものですが，FORTRANの場合はなかなかそうはいきません.しかし,このようなモジュールもあるものは市販されていますし,自作もそれほど困難ではありません.このような道具だてがどの程度揃うかで,実際の仕事の効率は大きく違ってきます.

科学技術計算では最も有力

今日，多くの種類のプログラミング言語があり，そのどれを選ぶかというのはさまざまな局面で問題になることが予想されます．その解答の基準は，何がしたいか？　ということだと思います．冒頭にも述べましたが，もしそれが科学技術計算に関わるものであるならば，そのもっとも有力な世界共通言語であるFORTRANを学ぶことは必須です．最後になりますが，パソコンやワークステーションの世界が大きく広がりつつあるこの時代にこそFORTRANをと思う方が1人でも増えることを願ってやみません.

プログラミング用語
セグメント　80x86プロセッサ独特のメモリを管理するための単位

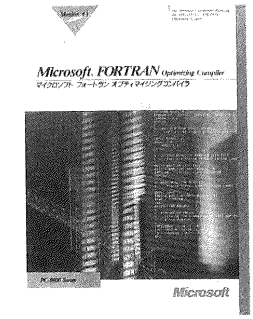
処理系紹介

FORTRAN

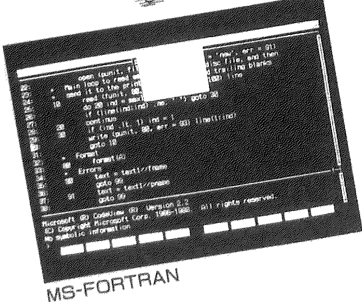
MS-FORTRAN

マイクロソフト社のFORTRANコンパイラ．ANSIのFORTRAN77規格に完全準拠，さらに独自の拡張を加え，IBM VSやDECのVAXの標準関数をも含む．MS-C 5.1で使われている高度な最適化機能を有し，高速でコンパクトなコードを作成する．MS-DOS用，OS/2用および両OS共用のプログラム開発が可能．ミディアム，ヒュージ，ラージの3つのメモリモデルをサポートする．マイクロソフト社の他の言語プロダクツ（MS-C 5.1, MASM5.1, QuickC, QuickBASICなど）で作成したプログラムとリンクして，ミクスドランゲージプログラムを作成することができる．Microsoft Editor，ソースレベルデバッガCodeView，オーバーレイリンカLINK，プログラム保守ユーティリティMAKE，ライブラリマネージャLIBなどの各種ユーティリティを含む．

NEC PC-9800シリーズ用，富士通FMRシリーズ用，沖電気if800／NTT BS/21用，日立B16/B32シリーズ用など各社パソコン対応版がある．



製品名：Microsoft FORTRAN Optimizing Compiler Ver.4.1
価格：80,000円
問い合わせ先：マイクロソフト（株）
03-3363-1201



MS-FORTRAN

Pro FORTRAN-77

英Prospero Software社のFORTRANコンパイラ．ANSI/ISO/JIS FORTRAN77規格に準拠しているため，ミニコンや大型機で開発されたプログラムをほとんど変更することなく利用できる．コンパイル，リンク，実行スピードが速く，コンパクトなコードを生成する．64KB～1MBの配列が利用できるので，大容量プログラムの作成も可能．シンボリックデバッガPROBE，高速オーバーレイリンカPro-LINK，多機能ライブラリアンPro-Libを含む．

NEC PC-9800シリーズ，富士通FMR-50/60/70およびFM TOWNS，松下Panacom M500/700/800，IBM PS55/5550/5560/5570，PC/AT，東芝J-3100/3300シリーズに対応．

製品名：Pro FORTRAN-77
価格：98,000円
問い合わせ先：（株）ライフボート
03-3293-4711

SW-Pro FORTRAN-77

英Prospero Software社のFORTRANコンパイラ．ANSI/ISO/JIS FORTRAN77規格に準拠．データサイズ，コードサイズとも64KB以内のスモールメモリモデル専用だが，付属のリンカによってオーバーレイが可能．シンボリックデバッガ，専用オーバーレイリンカ，専用ライブラリアンを含む．NEC PC-9800シリーズ用，FMRシリーズ用，松下Panacom-Mシリーズ用がある．

製品名：SW-Pro FORTRAN-77
価格：29,800円
問い合わせ先：（株）シェフロ
075-415-0885

Advanced RUN/FORTRAN

Pro FORTRAN-77をもとに作ら

FORTRAN

れたコンパイラと日本語フルスクリーンエディタ，対話型シンボリックデバッガPROBEのセット．プログラミング言語の学習環境を提供することを目的としたライフボート社のAdvanced RUNシリーズのひとつ．

インターフェースはBASICライクなコマンドベース．学習用とはいっても，ANSI FORTRAN77規格を満たしたフルセットのFORTRANコンパイラ．Pro FORTRAN-77とは完全な互換性がある．

NEC PC-9801シリーズ，富士通FMR50/60/70およびFM TOWNS，IBM PS55シリーズ，東芝J-3100シリーズに対応．

製品名：Advanced RUN/FORTRAN
価格：29,800円
問い合わせ先：(株) ライフボート
03-3293-4711

AWS-FORTRAN

absoft社製FORTRAN77コンパイラとMC68020+68882を搭載したアドオンCPUボード，ノーマル/ハイレゾ両用のグラフィックドライバ，MS-DOS拡張機能サポートソフト（SPARK-68K）のセット．プログラム，データサイズはメモリの許す限り無制限．ダイナミックリンク，VIRTUAL宣言が可能．68020，68882命令をインライン展開し，最適化したオブジェクトを出力する．グラフィックドライバを内蔵し，PC-9801のグラフィック機能が利用できる．Cやアセンブラとのインターフェースもサポート．ポジションインデペントコードの出力もできる．

ソースコードデバッガ，ライブラリソースが付属．

NEC PC-9800シリーズに対応．

製品名：AWS-FORTRAN System
価格：398,000円(16MHz)
498,000円(25MHz)
598,000円(33MHz)
問い合わせ先：カノーフス電子 (株)
078-411-5292

RM/FORTRAN

オーストラリアに本拠を置くAUSTEC社のFORTRANコンパイラ．ANSI FORTRAN77規格に完全準拠．VAX/VS FORTRANと同一の言語仕様をもつため，ミニコンやメインフレーム上のプログラムの移植が容易である．スクリーンエディタ，コンパイラ，リンカ，デバッガ，ソースコードマネージャの統合ツールであるRM/FORTE，8087/80287エミュレータ，高速構文チェッカーを内蔵．

NEC PC-9800シリーズ／N5200シリーズ，東芝J3100シリーズに対応．また富士通FMRシリーズ，松下Panacom Mシリーズ，三菱MAXYシリーズ対応版はそれぞれのメーカーより提供されている．

製品名：RM/FORTRAN Ver.2.6
価格：80,000円
問い合わせ先：(株) コミュニケーションサイエンス
03-3793-6751

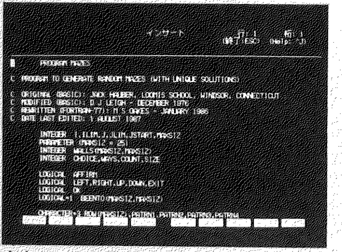
NDP FORTRAN 386

Microway社開発の80386用FORTRANコンパイラ．ANSI FORTRAN77規格に準拠したフルセットコンパイラ．80386CPUのプロテクトモードで動作するコードを生成する．実際のプログラム開発には別売のPharLap社の386|TOOLBOX（DOS|EXTENDER，プロテクトモード対応アセンブラ，リンカ，ライブラリアン，デバッガを含む）が必要となる．

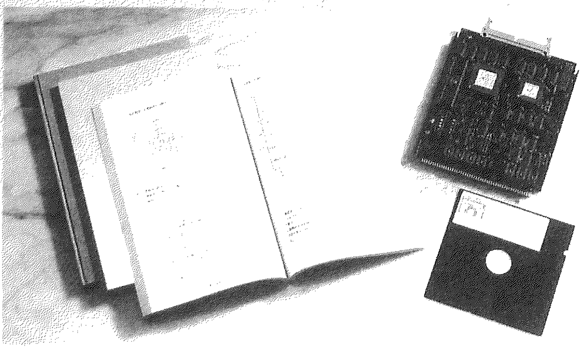
80386プロテクトモードで実行されるため，開発するプログラムの大きさには事実上制限がない．またMS-FORTRAN, R/M FORTRAN, VAX FORTRANからのプログラムの移植を容易にするための拡張機能をサポート．

NEC PC-9800シリーズの32ビット機，IBM PCとその互換機に対応．最低2MBの空き領域のあるハードディスクと640KB+2MB以上のプロテクトモード対応メモリが必要．数値演算プロセッサとして80287, 80387, mW1167, mW3167のいずれかが必須である．

製品名：NDP FORTRAN 386
価格：148,000円
製品名：386|TOOLBOX Ver.2.2
価格：148,000円
問い合わせ先：テックソフトアンドサービス (株)
0474-23-1322



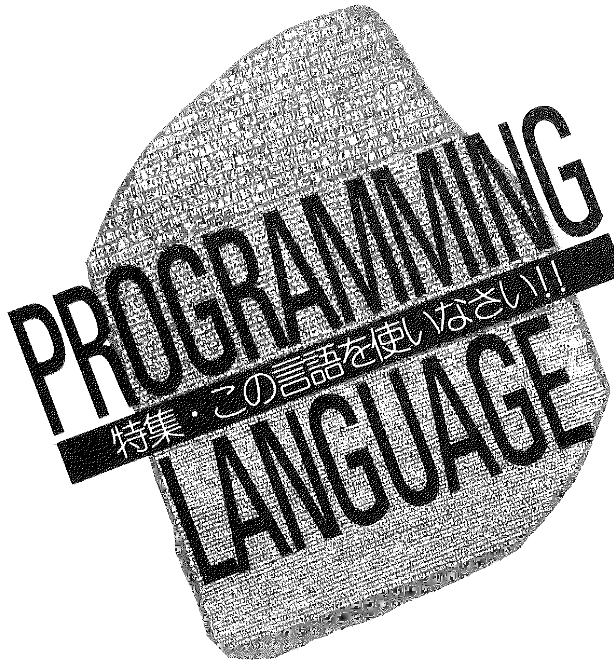
Advanced RUN/FORTRAN



AWS-FORTRANにはボードが付属している

処理系紹介

プログラミング用語
ソースファイル　プログラム本体を保持する単位で，ソースファイルを各処理系にかけることにより機械語の目的ファイルが作られる．



高度なソフトを容易に制作できる

HyperCard/HyperTalk

田村浩二

HyperCard登場

1987年8月，米アップル社はMacintosh用アプリケーションソフトウエアHyperCardを発表した．当時，Macintoshは「人に優しいコンピュータ」といって売られていたが，実用的に使った場合「MS-DOSより多少まし」な程度であって，やはり使いこなすにはたいへんな努力が必要であった．とくにプログラミングについては，その独創的な設計思想に基づくユーザーインターフェイス，ToolBoxと呼ばれるROM内ルーチンを理解しなければならず，一般の人はもとよりプログラマ達でさえも取りつきにくいというしろものだった．

また，MacintoshはPC-9801などのようにBASICを標準装備しておらず，一般ユーザーが簡単なプログラムを組みたいと思ってもまったく実現不可能であった．「何でもできそうで何にもできないMacintosh」などと陰口をたたかれていたものだ．

そこへ華々しく登場したのがこのHyperCardである．では，そのHyperCardとはいったいどんなジャンルのソフトウェアなのか．言葉ではいいあらわしにくいが，あえていうならばカード型データベースである．しかし，そのデータは単にテキストにとどまらず，グラフィック，アニメーション，サウンドなど従来のデータベースやアプリケーションでは扱いにくかった情報も扱え，しかもそれらの情報が互いに有機的にリンクしあっているのである．

HyperCardでの情報の単位はカードといい，このカードに文字，絵，音などの情報を記録する．そしてカード

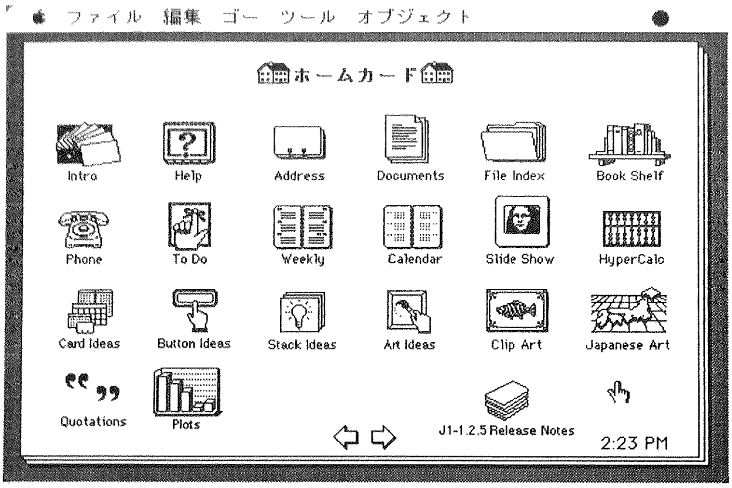


図1　HyperCardのホームスタック（ホームカード）の画面．ほかのアプリケーションスタックを起動するためのボタンが多数並んでいる．

を何枚か集めたものをスタックと呼び，このスタックがソフトウェアの単位となる．HyperCardの大きな特徴はこのスタックが「誰にでも簡単に作れる」ということである．そして，このHyperCardのスタックの動作を記述する言語がHyperTalkである．

HyperTalk

HyperTalkはSmalltalkの流れをくむオブジェクト指向言語である．しかし，対象がプロのプログラマではなくコンピュータでプログラミングをしたことがないような一般のユーザーであるためにビギナー向けに簡略化されてはいる．しかし，メッセージ継承，イベント駆動型プログラミングなどの特徴はそのまま残されている．

HyperTalkは，HyperCardを構成するオブジェクトに対してメッセージを送ることによって命令を伝え，実行していく．HyperCardのオブジェクトには，テキストデータを格納する「フィールド」，各種機能を実現する「ボタン」，フィールドやボタンを配置する「カード」，複数のカードで共有する「バックグラウンド」，すべてのカードをまとめる「スタック」，HyperCardのメインメニュー役をする特殊な「ホームスタック」，HyperCard本体，の7種類がある（図1）．これらのオブジェクトは自分にメッセージがやってきたときに何をすべきか，自分で処理できない場合には誰に聞けばよいのかを知っている．

たとえば，あるボタンにメッセージが渡ったとする．そのボタンがそのメッセージを受理してしまえばそれまでだが，自分で処理できない場合にはそのボタンが乗っているカードにメッセージを渡し，カードでだめならばバックグラウンドに渡り，次にスタック，ホームスタック，HyperCard，そしてHyperCardさえも受理できないような命令はエラーとなる．これがメッセージの継承である．メッセージの継承は，HyperTalkを使う上で重要な概念ではあるが，通常はそれほど意識する必要はない．

実際のプログラミング

以上でだいたいHyperCardとHyperTalkについて理解されたことと思うが，ここで簡単な例をあげてHyperTalkの具体的な使い方について説明していく．まず，メッセージを受け止めるメッセージハンドラの基本的な書き方は図2のようになっており，具体的には図3のように書く．

図3のメッセージハンドラがボタン

図2　メッセージハンドラの書き方

```
on  <メッセージ>   [<パラメータ>]
    命令1
    命令2
    ・・・
    命令n
end  <メッセージ>
```

図3　メッセージハンドラの例

```
on mouseUp
  play "piano"
  put "Welcome to Macintosh" into the message box
  visual effect scroll left very fast
  go to next card
end mouseUp
```


プログラミング用語
多岐選択文　制御文の一種で，switch～文のように複数の条件に対し，それぞれ実行される文．

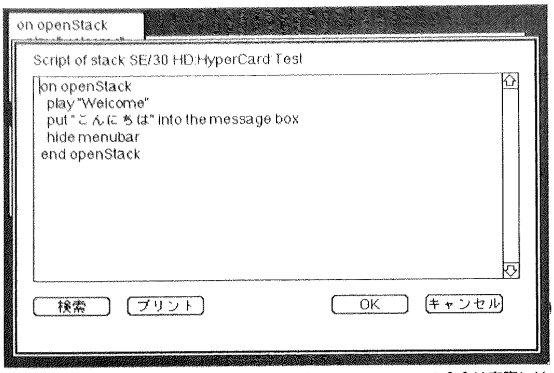


図4 サンプルスタックの画面．枠で囲ってあるHyperTalkの命令は実際には画面には表示されない．

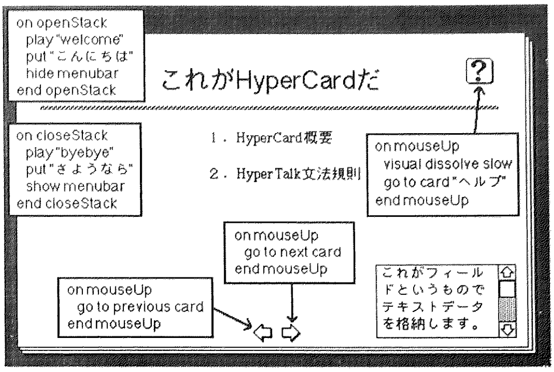


図5 HyperTalkスクリプトの編集画面．あらゆるオブジェクトに対しこのように命令を与えることができる．

に書いてあったときの意味は，「もしボタンの上でマウスのボタンがはなされたら"piano"という音を演奏し，"Welcome to Macintosh"とメッセージボックスに表示し，素早く左にスクロールしながら次のカードへ行け」ということである．

HyperTalkの文法は英語の文法に非常に近いので，HyperTalkをまったく知らなくてもプログラムの意味を理解することができるし，英語圏の人であれば素人でも子供でも容易に覚えることができる．また，HyperTalkの命令セットに不満が出てきたら，ほかのコンパイラ言語で作成した独自の外部コマンド・外部関数などを容易に追加することもできる．Pascalなどほかの多くの言語にも関数定義などの機能があるが，HyperTalkの場合，外部コマンドはMacintosh特有のリソース構造になっており，マシンコードの状態でほかのスタックへの移動が可能である．これにより，ソースリストが公開されていない場合でも他人の優秀な外部コマンドを使用することができ，入門者でも実力以上のハイレベルなスタックを作ることができる．

しかし，HyperTalkはけっして初心者専用の言語ではない．上級者は低レベルルーチン制作などの無駄な労力を省くことができ，しかもHyperCardのユーザーインターフェースを利用した他人にわかりやすいソフトウェアの制作が容易に行える．よって，自分が作りたいソフトウェアの本質的な部分だけを考えていればよく，ソフトウェアの生産性からみても精神衛生上からもほかの開発環境よりも優れているといえるだろう．

HyperTalkで組め！

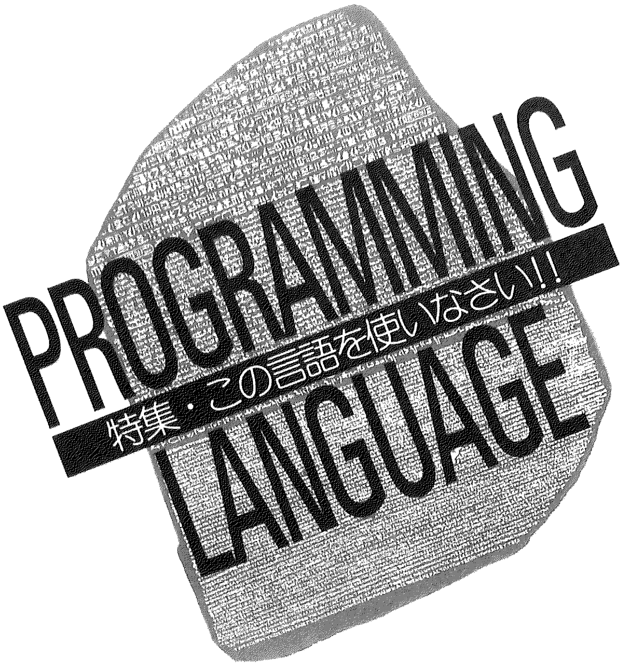
このように初心者から上級者まで幅広く対応が可能なHyperTalkであるが，やはり向き不向きというものがある．基本的にインタプリタなので，複雑な処理や高速な計算を要求するものには向いていない．逆に，高速検索機能が使えるデータベース，グラフィックスとマンマシンインターフェースが重要になるプレゼンテーションなどにはうってつけである．この辺りはHyperCardの機能からして当然かもしれないが，紙芝居のようにただ紙をめくるだけのようなソフトの制作を考えた場合，HyperCardならばものの10分もあれば完成してしまうが，ほかの処理系では，グラフィックの表示・スクロール処理，マウス処理，サウンド処理などサブルーチンの準備だけでも数時間はかかってしまいそうだ．

とにかくHyperCardほど説明しにくいものはないので，一度でもよいから実際に触ってみてほしい．そしてその先進的で洗練された環境で実際に何か作ってみてほしい．とくにMS-DOSでグラフィックアドベンチャーなどを作っている連中にいいたい．HyperTalkで組め！と．

HyperCard 2.0

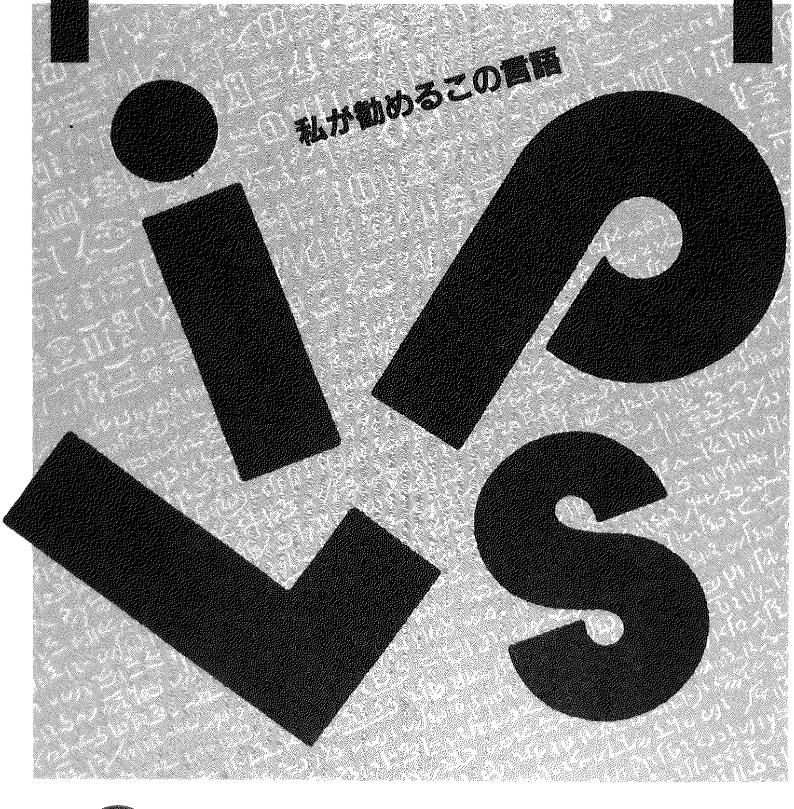
1987年の発表以来，HyperCardはMacintoshのプログラミングプラットホームとして多くのユーザーから支持されてきたが，同時にHyperCardの機能不足に対する不満も数多く指摘された．この不満を解消するべく開発・発表されたのが，HyperCard 2.0である．HyperCard 2.0では，マルチスタックと可変カードサイズに対応し，デバッガやアイコンエディタなどのツール類も加えられた．また，HyperTalkも多くの機能強化が図られ，スクリプトでメニューをカスタマイズする機能などが追加されている．発売時期，価格などは未定．問い合わせは，アップルコンピュータジャパン（株）（TEL 03-5562-6210）まで．

プログラミング用語
単岐選択文　制御文の一種で，if～文のように条件が成立する時のみ実行される文．



高速・大容量時代に能力を発揮する

Lisp　　Lambdd Software Works　藤沢　滋



最近までのAIブームのお蔭でLispという処理系が一般に知られるようになったが，パーソナルユースのよい処理系がなかなか出なかった．ところがここ2，3年パソコンの高速化，メモリの大容量化が著しいのに併せてSchemeというLispの変種で極めてシンプルでかつ強力な処理系が台頭してきており，メモリ喰いのLisp系言語も十分なパフォーマンスを発揮できるようになってきた．

以下にこの言語のあらましを示し，勧める理由としたい．要領を得ぬところがあれば浅学故にご容赦願いたい．

Lispの生い立ち

Lispは，1950年代末から1960年代初頭にかけての人工知能研究用として，MITのJ. McCartyらにより生み出された関数型言語で，FORTRANと同じくらいに由緒ある言語である．

この言語は文字よりも記号を扱うことを考えて設計されており，知識表現を扱いやすいようにリストという動的な構造を取り込んでいる点が大きな特徴となっている．Lispという名前はLISt Processorからきているのである．

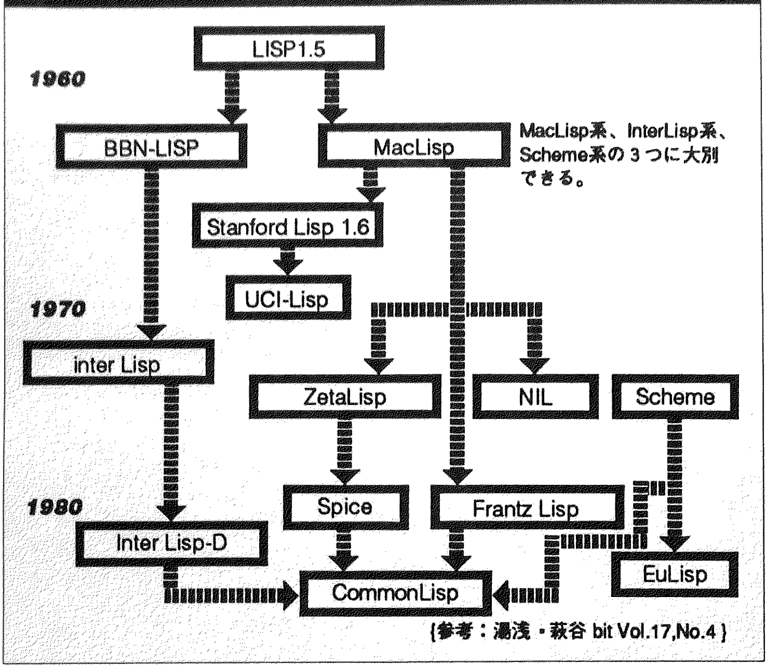
また，Alonzo Churchの提唱したλ計算に基づく機能を取り込んでいるため，関数（作用）をデータとして扱うことができる．

また，かなり初期の頃からコマーシャルユースではなく研究室内で内輪に使われながらハックされてきており，ソースを記述するための単なる記述言語としてではなく，デバッガやインスペクタを含む「作業環境」として発展してきた点がほかの言語と大きく異なっている．

ほとんどのLispでは，ほかの言語に

プログラミング用語
データ型　voidを含まない型指定修飾子の並び，関数以外の直接抽象演算子である．

図：Lispの系譜



おける構造体やベクトル，配列などはもちろん使えるし，さらに以下の特徴をもつ．

Lispの特徴

リスト

大きさに制限のない１次元配列のようなもの．要素を自由に継ぎたしたり取り除いたりできる手段が言語レベルで保証されているのはありがたい．

λ表記

名前のない関数，つまり作用だけが意味をもつ関数を記述するのに使われる．関数引数に便利．Churchのλ計算をもとに作られた機能．

たとえばCのqsortを使うとき，１回しか使わなくても大小判定用の関数に名前を与えなければならないが，Lispの場合はlambda表記で局所的に関数を記述できる（リスト１の２つ目の実行例参照）．

アトムに基づく入出力機能を言語レベルでサポート

文字レベルの入出力もあるが，より高度なアトムの入出力もシステムがサポートしている．

区切り記号(通常はtabや改行，スペースなどの空白文字）をもつデータの入出力が容易．

リストによるシンボル管理機構と合わせてコンパイラのパーサ作成や，英語のような空白で語が区切られるタイプの自然言語の構文解析部の作成に向いている．

型制約が極めて弱い

データ自身が型情報をもっており，コードが型に依存しにくい．最近の処理系では逆に型宣言を導入して，メモリ効率や速度を上げるものもある．

また，以下の機能は必ずしもすべてのLisp処理系にあるわけではないが，Lispを特徴づけるものであるためあげておく．

多倍長整数(bignum)，分数(rational)

事実上無限桁の整数演算および分数演算のサポート．これにより桁落ちなしの計算ができる．数式処理にたいへん役にたつ機能である．

インクリメンタルコンパイル

Lisp処理系下では，インタープリタモードの関数とコンパイルされた関数が混在できる．完成した関数から順次個別にコンパイルしておき，デバッグ中のものはインタープリタモードで走らせることができる．

コンパイラでありながらリンケージ操作がない．逆に，OS下で直接実行可能なバイナリを生成できる処理系は少ない．コンパイルした関数を走らせるには，その処理系のカーネルが必要．

構文マクロ

マクロが非常に強力で，言語のもつ構文をどんどん拡張することができる．ほかの処理系からプログラムを移植するとき，若干の構文の違いや欠如はマクロで吸収できる．

統合環境

Lisp 処理系は通常単なるコンパイラや実行系としてではなく，統合環境として提供されることが多い．このため，エディタの他Hook機能やステッパなどのデバッガが用意されており，インクリメンタルコンパイルと合わせて必要な部分だけトレースしたりコンパイルしたりできる．

クロージャ（関数閉包）

この機能により，特定の関数間でのみ相互参照可能なシンボルを管理できる．この機能を使うとオブジェクト指向プログラミングのクラス構造を自然

低水準関数　ストリームを介さない，ファイルの入出力などハードウェアに密接した関数を差す，できが悪い関数を差すわけではない．

な形で構築できるため，オブジェクト指向型のシステムが容易に提供できる．

本稿では省略したが，このほかにも動的束縛やSchemeの遅延束縛，継続(コンティニュエーション)，Common Lispのキーワード引数などのたいへん便利で強力な機能がある．

サンプル

動的な構造であるリストを使ってクイックソートのプログラムを示す（リスト1）．

2番目の例では，qsort中でworkは数ではなくリストになる．型制約が弱いので，このような型に依存しないコーディングが可能となる．

結論

最近はパソコンでさえ80x86や680x0など，高速な上に非常に巨大なメモリ空間が利用できるようになってきた．今までのようにメモリや速度を気にしながらものを作る時代は終りつつあるといえよう．

また，各種インターフェイスにバイトレベルの低水準のものから，より高水準の記号入出力を要求されるものが増えてきている（自然言語インターフェイス，計算機言語コンパイラのパーサ部分など）．

これらをフレキシブルな環境で行使してプロトタイプを作成し，仕様を検討修正のうえ本物を作るほうが時間的にも金銭的にも軽減されるであろう．そのためにはよいプロトタイピング用の環境を整える必要があるが，Lispはこれらの要求に応える絶好の言語である．これをもって本論でLispを勧める理由としたい．

リスト1

```
;;  seq はソートされる要素の並び（リスト）
;;  pred は2引数を与え，両者の関係を調べる述語（真か偽を返す関数）

(defun qsort (seq pred)
  (if (null? seq)
     '()            ;; 再帰呼出の終了条件
     (let ((pivot (car seq)) ;; 使用する変数の宣言と初期化
           (group1 nil)
           (group2 nil)
           (work nil))
       ;; pivot（軸）を境に2つのグループに分ける
       (do ((rest (cdr seq) (cdr rest))) ((null? rest) nil)
         (setq work (car rest))
         (if (funcall pred work pivot)
           (push work group1)
           (push work group2)))
       ;; それぞれのグループをさらにソートし，結果を結合する 【注1】
       (append (qsort group1 pred) (list pivot) (qsort group2 pred)))))

注1： 要素の並べ替えがこの1行に集約されている点に注目されたい．

> (qsort '(1 20 4 -1 0) #'>)
(20 4 1 0 -1)

> (qsort '((佳世子 (age 18)) (みゆき (age 20)) (由佳 (age 19)))
        #'(lambda (a b) (> (cadr (assoc 'age (cdr a)))
                           (cadr (assoc 'age (cdr b))))))
((みゆき (age 20)) (由佳 (age 20)) (佳世子 (age 18)))
```

参考文献

[0] はじめての人のためのLISP，竹内郁雄 著，サイエンス社 ソフトウェアライブラリ3
[1] LISP，P. H. Winston，B. K. P. Horn 著，白井良明・阿部憲広 訳 倍風館 情報処理シリーズ4
[2] LISP，P. H. Winston，B. K. P. Horn，Addison-Wesley，[1]の原書
[3] Scheme手習い，D. P. Friedman，M. Felleisen 著，元吉文男・横山晶一 訳 マグロウヒル出版
[4] The Little LISPer，D. P. Friedman，M. Felleisen，The MIT Press，[3]の原書
[5] Anatomy of LISP，J. Allen，McGraw-Hill
[6] Common LISP The Language，G. L. Steele Jr.，Digital Press
[7] Common Lisp 入門，湯浅太一・萩谷昌己，共立出版 bit vol. 17 No. 4～6
[8] Revised³ Report on the Algorithmic Language Scheme，H. Abelson，et. al. MIT AI Lab. Memo 848a，Sep. 1986.（ACM SIG PLAN Noteces，Dec. 1986 にも）
[9] 日本語PC Scheme リファレンス・マニュアル，センチュリ リサーチセンタ（株）
[10] Lispの理論的背景，戸島熙，CQ出版 アーカイブ Nov. 1986 pp. 8～37

[0]は入門書を装っているが各所にちょっとしたネタが仕込んであり，読んで笑えるようになれば一人前のLisperである．
[1]，[2]は実例に即した例題が豊富な解説書で，Lispそのもの以上にアルゴリズムのネタ本として使える．仕事でお世話になった人は多いと思う．:-) なお，原書のほうは改版してCommon Lisp対応になった．
[5]は自分で処理系を作りたいとか，Lispを究めたい人は必読の書である．硬派のバイブル．

プログラミング用語
テキストモード　ファイルに対して文書などのテキストを読み書きする．0x0d,0x0aを改行し，0x1aをファイルの終端と判断する．

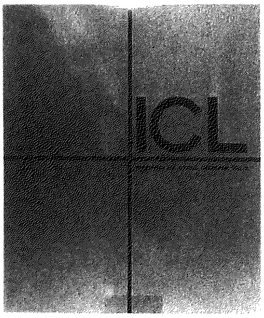
処理系紹介

Lisp

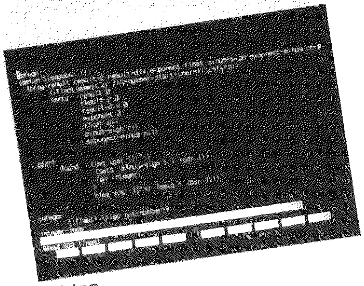
ICLisp

Common Lisp準拠のLisp処理系。コンパイラ版とインタプリタ版の2種類がある。コンパイラ版はインタプリタを含む。インタプリタ版はコンパイル版からコンパイラ機能，無限多倍長整数演算，有理数演算，浮動小数点演算機能を除いたもの。メモリ実装640KBのフロッピーベースの構成でも運用できるコンパクトな処理系。

NEC PC-9800シリーズ，富士通FMRシリーズ，16β，IBM 55シリーズ，IBM PC，東芝J3100シリーズを始めとする各社パソコンに対応。ただし，コンパイラの実行にはV30, 80186, 80286, 80386, 80486のいずれかのCPUを搭載している必要がある。



製品名：ICLisp V.4.11 コンパイラ
価格：98,000円
製品名：ICLisp V.4.11 インタプリタ
価格：60,000円
問い合わせ先：㈱インフォメーション・アンド・コントロール研究所
03-3352-4746

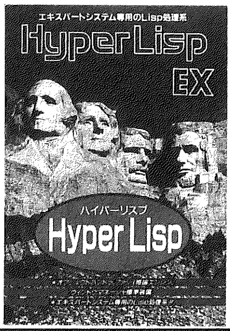


ICLisp

Hyper Lisp

オブジェクト指向型エキスパートシステム構築ツール「Hyper-Brain／脳力男」の記述言語として開発されたLisp処理系。MacLispとCommonLispの言語仕様を参考に，さらにオブジェクト指向型プログラミング環境，ファジィ推論エンジン，ビットマップウィンドウシステムによるユーザーインターフェース，画像表示機能，日本語音声出力機能を追加している。

NEC PC-9800シリーズに対応。80286以上のCPU搭載機種のプロテクトモードで動作する拡張版のHyper Lisp EXもある。



製品名：Hyper Lisp
価格：49,000円
製品名：Hyper Lisp EX
価格：147,000円
問い合わせ先：㈱ブレインズ
03-3729-8151

GCLISP

GCLISP（Golden Common Lisp）は，米Golden Hill Computers社製のLispインタプリタ＆コンパイラ。Common Lisp準拠。Lattice Cで作成したオブジェクトとリンクが可能。Lispの初心者向けCAIシステム日本語サンマルコエクスプローラ，Lisp専用エディタGMACSを含む。

動作には80286以上のCPUと本体メモリ3MB以上を備えたNEC PC-9800シリーズ，ハードディスクに6MB以上の空き容量を必要とする。

富士通FMRシリーズ，IBM PS/55シリーズ対応版もある。

製品名：GCLISP 286 DEVELOPERV2.4J PC-9800シリーズ用
価格：476,000円
問い合わせ先：㈱シーイーシー
03-3407-7481

It's CL/386

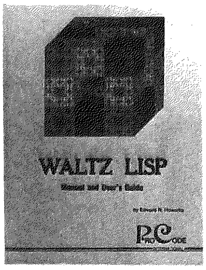
日本語フルセットCommon Lispインタプリタ＆コンパイラ。NEC PC-9800シリーズの80386CPU（含む80386SX）搭載機種で動作。メインメモリは2MB以上必要。

製品名：It's CL/386
価格：148,000円
問い合わせ先：㈱ヒー・ユー・シー
011-807-6621

Lisp

Waltz LISP

Pro Code International社のLispインタプリタ．同社のFrantz-Lispコンパイラのサブセット．10MHz80286CPUのMS-DOS上でVAX/780のFrantzLispとほぼ同じ処理速度を実現している．Lisp専用の構造化エディタがライブラリとして付属するほか，Wordstar系コマンドの汎用エディタも組み込まれている．サンプルとしてLispの関数として記述されたPrologインタプリタが付く．英文のマニュアルのほかに，Prologマニュアル，和文の手引きが別冊で付属．



製品名：Waltz LISP Ver.5.12
価格：39,800円
問い合わせ先：㈱ササンパシフィック
045-314-9514

Advanced RUN/LISP

Common Lispに準拠したLispインタプリタ．最大8MBの仮想記憶をディスク上に確保してメモリ容量の限られたMS-DOSパソコン上での開発を可能にしている．プログラミング言語の学習環境を提供することを目的としたライフボート社のAdvanced RUNシリーズのひとつ．

NEC PC-9801シリーズ，富士通FMR50/60/70，FM TOWNS，東芝J-3100シリーズ，IBM PS/55シリーズに対応．

製品名：Advanced RUN/LISP
価格：29,800円
問い合わせ先：㈱ライフボート
03-3293-4711

muLISP87

Common Lisp準拠の512個の関数を備えたLispシステム．日本語マニュアル付き．最低128KBのメモリで動作．コンパイラとインタプリタを含むフルセットとインタプリタのみの製品がある．

NEC PC-9800シリーズ用，富士通FM16/Rシリーズ，東芝J/T-3100，IBM5550/60シリーズ，AX互換機に対応．

製品名：muLISP87 V.6.20 フルセット
価格：74,900円
製品名：muLISP87 V.6.20 インタプリタのみ
価格：49,900円
問い合わせ先：㈱ソフテック
0480-85-6565

XLISP

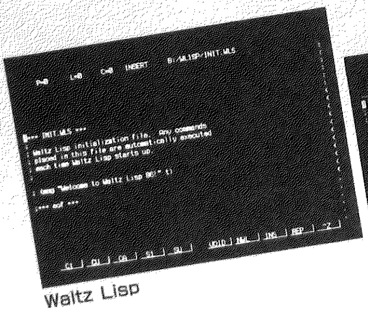
小型コンピュータ上でオブジェクト指向プログラミングとLispの学習・実験の環境を構築するためにDavid Michael Betz氏によって開発された．Common Lispにオブジェクト指向の機能を付加した仕様となっている．現在では，IBM-PCとその互換機，Macintosh，Amiga，Atari-STなど各種マシン用がある．XLISP自体はC言語のみを使って記述されており，ソースも公開されている．営利を目的としないユーザーなら，無料でソースを利用，変更することができる．付属のドキュメントはXLISPの文法を紹介する簡潔なもの．Lispそのものやオブジェクト指向プログラミングの概念を学ぶには，他の参考書と併用することがすすめられている．

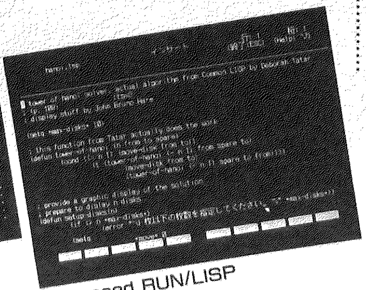
製品名：XLISP An Object-oriented Lisp Ver.2.0
問い合わせ先：David Michael Betz
(603) 924-6936
U.S.A.

XSCHEME

XLISPと同じくDavid Michael Betz氏によって開発された．Lispの変種であるSchemeにオブジェクト指向の機能を付加した仕様となっている．IBM-PCとその互換機，Macintosh，Amiga，Atari-STなど各種マシン用がある．XLISP同様C言語のみを使って記述されており，ソースも公開されている．

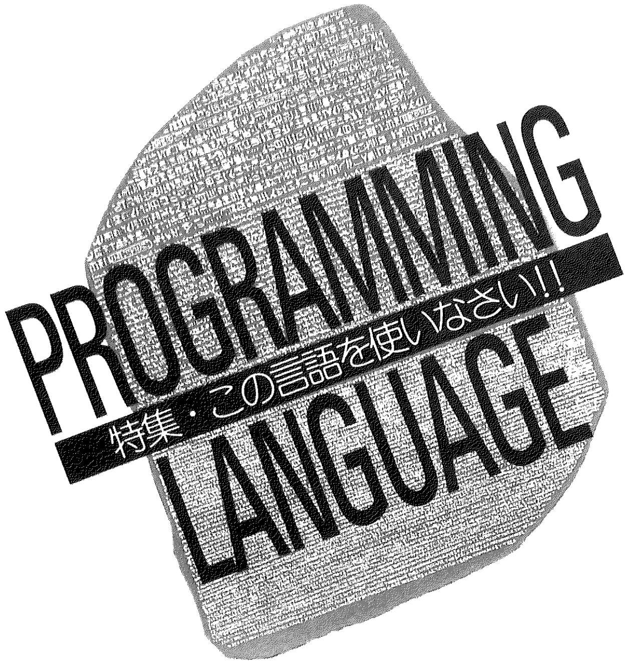
製品名：XSCHEME An Object-oriented Scheme Ver.0.15
問い合わせ先：David Michael Betz
(603) 924-4145
U.S.A.


Waltz Lisp


Advanced RUN/LISP

処理系紹介

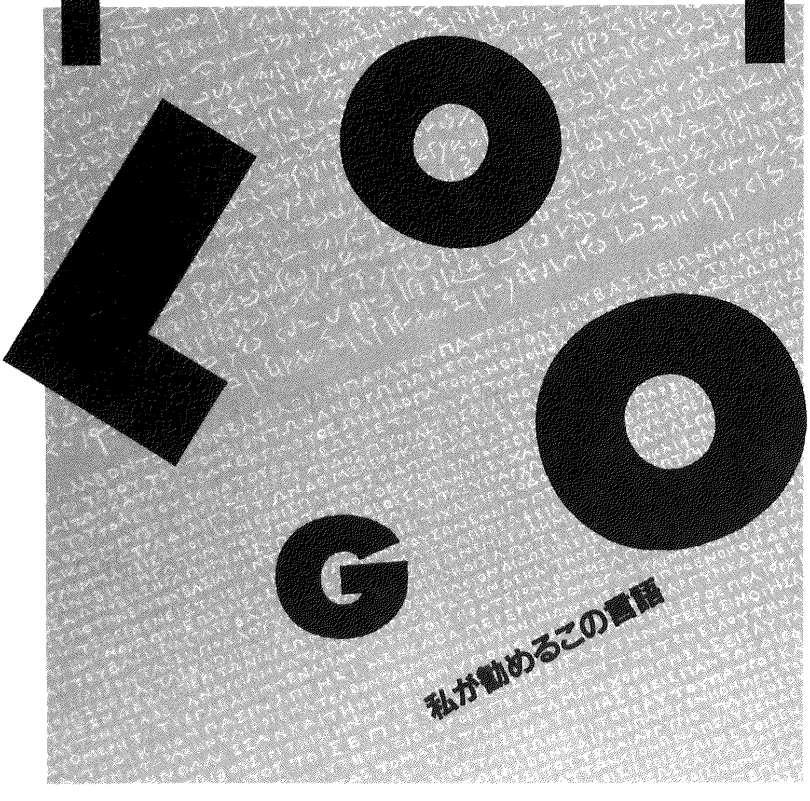
プログラミング用語
デバッグ　プログラムの誤りを訂正する作業．デバッガを使わず，目でプログラムの流れを追いながらする作業をハンドデバッグという．



効率より学習の容易さを優先

LOGO

足利高等学校定時制教諭 福田 洋



7歳の子供が使えるように

子供の食べ物の好き嫌いは，学習によるものが大きいという．たとえば，日本人の子供の多くは,「ああ，うまい，オイシイ」という両親のことばを聞きながらはじめて味噌汁を飲む．すると，味噌汁とオイシイがbindされて［味噌汁　オイシイ］というリストが子供の頭のなかにpushされて，以後，味噌汁をのむと，オイシイという気持が呼び出されることになる．

言語の好き嫌いについても，それに最初に接した環境が大きく作用すると思われる．CRT画面に向かって，FD 100とタートルを100歩動かすことを楽しいと感じるのは，やはり圧倒的に子供だろう．LOGOを使っている学校では，そのことで子供は，ほめられることはあっても，おこられることはまずない．そして子供はLOGOのファンになる．

ところが，コンピュータメーカーやソフト会社で，技術者が，FD 100とやっていると上司から「なにを遊んでいるのだ」とおこられてしまうかもしれない．すると，［LOGO　オコラレル］というリストが技術者の頭の中にpushされLOGOのファンが１人減ってしまうことになる．

しかし，もし，大人も仕事と関係なく子供のように遊ぶことができるとすれば，タートルを動かして絵を描くことを楽しいと感じる人は多いのではないかと思う．

日本語のすぐれた特徴の１つは，〝ひらがな〟をもっていることだろう．７歳の小学校１年生は，たった１人で日記や手紙を書くことができる．大人は日本語で，よい詩やすぐれた小説，論

文も書く.

言語が次の世代に伝わるためには，子供に理解できなくてはならない．子供が使えることは，言語にとってごく当然の必要条件なのである.

LOGOの父，S・パパートは，コンピュータ言語についても，わかりやすさや子供が使えることが重要であることを繰り返し説いている．LOGOはまさに，7歳の子供が日記や手紙が書けるようにと意図された言語である．タートルグラフィックスは，そのために大きな役割をしている．タートルは，はじめ，子供のために用意されたものだが，最近では，いろいろな方向に進化しはじめている.

LISP系言語としてのLOGO

次に，LOGOのもう1つの側面，LISP系言語としてのLOGOということについてお話したい．そのまえに，

問題1　LOGOの母はだれか（注：ママートではない）

ついでに，

問題2　LOGOの祖父はだれか（注：ピアジェは父方）

最近「心の社会」の日本語訳がでたミンスキーは，若い頃，パパートとともにパーセプトロンの研究をしている．また，情報理論や人工知能で有名なシャノンがMITで助手として採用した2人がミンスキーとマッカーシーである．マッカーシーは，LISPの作者であるから，LOGOとLISPのつながりの深さもうなずける．LOGOはLISPを母体としているので，言語上の特徴の多くはLISPのものである．しかし，けっしてLISPにとらわれることなく，いろいろな改良のあとがみられる.

LISPでは，(a (b c d)) と書くと，aが関数名で (b c d) は引数，bは関数名でc，dは引数とわかる．LOGOでは，これはa b c dと書ける．どちらがよい記法かは意見のわかれるところだと思うが，()が複雑に入り組む構文のときは，LOGOで書くのは楽である．また，IF THEN ELSE文が

TMSのプログラムリスト

```
TO TMS
 init
 REPEAT 9999                                  |
    [PR :Ids                                  |
     IF (LAST :Ids = :Qh) [PR [ｵﾜﾘ] STOP]|
     MAKE "Ids ONESTEP :Ids]
END

TO ONESTEP :ids
 LOCAL [tape tapeji jyotai muki nxtmov]
 MAKE "tape FIRST :ids
 MAKE "tapeji WATCH :tape
 MAKE "jyotai LAST :ids
 MAKE "nxtmov nmf :jyotai :tapeji
 MAKE "tape REWRITE :tape FIRST :nxtmov
 MAKE "muki LAST :nxtmov
 IF (:muki = "R) [MAKE "tape RGTMOV :tape]
 IF (:muki = "N) [MAKE "tape NOMOV :tape]
 IF (:muki = "L) [MAKE "tape LFTMOV :tape]
 MAKE "jyotai FIRST BF :nxtmov
 OP LIST :tape :jyotai
END

TO WATCH :t
 IF LIST? FIRST :t [OP FIRST FIRST :t]
 OP WATCH BF :t
END

TO REWRITE :t :new
 IF EMPTY? :t [OP []]
 IF LIST? FIRST :t                            |
    [OP FPUT FPUT :new [] BF :t]
 OP FPUT FIRST :t                             |
        REWRITE BF :t :new
END

TO NOMOV :t
 OP :t
END

TO RGTMOV :t
 IF EMPTY? :t [OP []]
 IF LIST? FIRST :t                            |
    [OP FPUT FIRST FIRST :t                   |
            FPUT FPUT FIRST BF :t []          |
                 BF BF :t]
 OP FPUT FIRST :t RGTMOV BF :t
END

TO LFTMOV :t
 IF EMPTY? :t [OP []]
 IF LIST? LAST :t                             |
    [OP LPUT LAST LAST :t                     |
             LPUT LPUT LAST BL :t []          |
                  BL BL :t]
 OP LPUT LAST :t LFTMOV BL :t
END

TO init
 PR [2 * 3 = 6]
 MAKE "Qs "A
 MAKE "Qh "K
 MAKE "Ids [[[1]1 1 B 1 1 1 1 B B B B B  |
        B B B B B B]A]
END

TO nmf :q :s
 IF AND (:q = "A) (:s = "1) [OP [B B R]]
 IF AND (:q = "B) (:s = "B) [OP [B J R]]
 IF AND (:q = "B) (:s = "1) [OP [2 C R]]
 IF AND (:q = "C) (:s = "B) [OP [B D R]]
 IF AND (:q = "C) (:s = "1) [OP [1 C R]]
 IF AND (:q = "D) (:s = "1) [OP [1 E R]]
 IF AND (:q = "E) (:s = "B) [OP [B I L]]
 IF AND (:q = "E) (:s = "1) [OP [2 F R]]
 IF AND (:q = "F) (:s = "B) [OP [B G R]]
 IF AND (:q = "F) (:s = "1) [OP [1 F R]]
 IF AND (:q = "G) (:s = "B) [OP [1 H L]]
 IF AND (:q = "G) (:s = "1) [OP [1 G R]]
 IF AND (:q = "H) (:s = "B) [OP [B H L]]
 IF AND (:q = "H) (:s = "1) [OP [1 H L]]
 IF AND (:q = "H) (:s = "2) [OP [1 E R]]
 IF AND (:q = "I) (:s = "B) [OP [B I L]]
 IF AND (:q = "I) (:s = "1) [OP [1 I L]]
 IF AND (:q = "I) (:s = "2) [OP [B B R]]
 IF AND (:q = "J) (:s = "B) [OP [1 K N]]
 IF AND (:q = "J) (:s = "1) [OP [B J R]]
 PR [nmf ｶﾞ ﾀﾀﾞｼｸ ﾃｲｷﾞｻﾚﾃｲﾏｾﾝ ] OP []
END
```

プログラミング用語
日本語処理　全角文字は2バイトで表現されるため，日本語文字列操作では特別な関数に操作や判断が必要な場合がある.

使え，四則演算では中置法も使える．適当に () や字下げをいれて書けば，読みやすさも保てる．

LISPのCARにあたるFIRSTは，引数にはリスト，ワード，数のいずれでもよく，めんどうな判断はインタープリタがする，ということがLOGOの考え方である．つまり，効率よりもわかりやすさ，学習の容易さを優先させているわけである．

チューリングマシンシミュレータ

LOGOのプログラム例として，チューリングマシンシミュレータ（TMS）のプログラムを紹介する．リストと関数を主体にしたLOGO言語の特徴によって，このような問題のプログラムには威力を発揮する．

数学上のたいていの問題は関数によって定義されているから，関数ごとにLOGOのprocedureを作ってゆけばよい．実行されてゆく過程をタートルを使って表示するなど，いろいろと想像をふくらましてゆくこともできる．

TMSの実行例

```
2 * 3 = 6
[[1] 1 1 B 1 1 1 1 B B B B B B B B B B] A
[B [1] 1 B 1 1 1 1 B B B B B B B B B B] B
[B 2 [1] B 1 1 1 1 B B B B B B B B B B] C

      ......

[B B B B B B B B [1] 1 1 1 1 1 1 B B B B] K
ｵﾜﾘ
```

チューリングマシンを動かすにはTMSとする．テープはリストで表し，ヘッドが眺めている記号は [] で囲う．また，そのときのヘッドの状態はテープの右はじに書くことにする．nmfは次の動作を規定する関数（next move function）で，ここはTMSの使用者が作る．この例では，かけ算をする機械を作っている．

改良され発展しつつある道具

ここ栃木県足利市や，となりの群馬県太田市，桐生市は両毛地区と呼ばれ，「かかあ天下と空っ風」の地域として知られている．なぜ「かかあ天下」になったのかというと，それはこの地域で発展した繊維産業に原因がある．主婦たちは，機（はた）を織り，あるいは家でミシンを踏んで家計を助けた．そして，経済的な基盤ができて「かかあ天下」といわれるようになった．

LOGOのこれからの課題は，ミシンのようにそれによって多くの人が生きてゆけるようになることだと思う．そのためには，まず，よいコンパイラが作られ，アプリケーションプログラムが.EXEファイルとして独立して生成される必要がある．

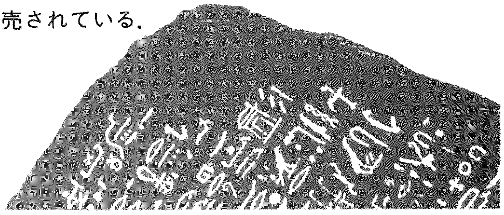
あるいは，コンパイラと同等の性能をもつインタープリタが開発され，どのコンピュータにもLOGOシステムが乗っているという必要がある．LOGOは，けっして完成された教育用言語ではなく，改良され発展しつつある1つの道具である．そこがまたおもしろいのである．

SNOBOL

1960年代初めにアメリカのベル電話研究所において開発された．SNOBOL1，SNOBOL2…と改良されて，現在はSNOBOL4が標準SNOBOL処理系となっている．SNOBOLは基本的にFORTRAN，COBOL，PASCAL，PL/Iなどと同じ手続き型言語であるが，他のどの言語と比較しても文字列処理の機能が充実している．

SNOBOLとは「StriNg Oriented SymBOLic」から取り出した文字を組み合せた言葉で，他の多くの言語のように「数」を処理するのではなく，「テキスト」を処理するために設計された言語である．このため，とくに人文科学研究への応用が期待されている．

SNOBOLの最大の特徴は，文字列を分析するためのパターンマッチの機能がきわめて強力である．ユーザーが定義したパターン変数と，組み込みのパターン関数を組み合せることで複雑なパターンマッチ処理が数行で実現できるのである．またデータ型が豊富なこともあげられる．テーブルという柔軟性の高いデータ型や，ユーザーの定義によって動的なデータ構造も実現することができる．他にも変数の間接参照や，変数の名前による引渡し，式の評価の引き延ばしなどもあげられる．現在はIBM PCやMacintosh，そしてNEC PC-9800シリーズなどのパーソナルコンピュータ用として販売されている．



プログラミング用語
ネスト　複文・反復制御・選択制御構造・条件組み込みなどに存在する．ソースプログラムでは目で判断できるようインデントを行う．

処理系紹介

LOGO

ロゴライター

ロゴライターは，Logoをベースに開発された教育用コンピュータソフトである．Logoのタートルグラフィックスに日本語ワープロ機能が加えられ，Logoで描いたグラフィックスに日本語の文章を書き込んだり，グラフィックスと文章を印刷することが可能となっている．

Logoの基本コマンドの入力は，日本語（ひらがな・漢字）と英語のどちらでも可能で，さらに日本語と英語を混在させることもできる．

ロゴライターで作成したグラフィックス・文章・プログラムは，ページと呼ばれるファイルで管理され，目次から開きたいページを簡単に選択できるようになっている．ページは，コンピュータが作業をし結果を表示する「ページの表側」とプログラムを記述する「ページの裏側」に分かれている．

また，各ページごとに150個まで形（シェイプ）を作成・登録できるため，アニメーションなどを簡単に実現することができる．

ロゴライターには，小学校キット，小学校学年別辞書，「情報基礎」ボックス，中学校教科学習ボックス，サンプルプログラム集，ツール集，シェイプ集Ⅰなど，数多くの関連製品が発売されている．

また，1991年４月には，CEC仕様'90に準拠したロゴライター２へのバージョンアップが予定されている．

製品名：ロゴライター
価格：43,000円
問い合わせ先：ロゴジャパン㈱
03-3470-6511

3D-LOGO

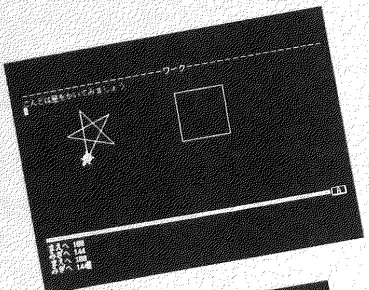
3D-LOGOは，たんなるLogo処理系というよりグラフィックス作成ソフトウェアというべきものである．

32000×32000×32000という大きさの空間に，64匹のサイバネティック・アニマルータートルを動かすことで自由にコンピュータグラフィックスを作り出すことができる．タートルは３次元の姿をもち，その形を自由に変えることができるので，タートルをキャラクタとしてコンピュータアニメーションを作ることも簡単に実現可能である．

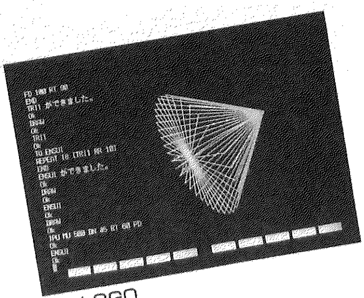
また，タートルは目をもっており，この視点を動かすことで，映像のズームアップや複雑な立体図形をさまざまな視点から観察したり，複数の視点から見た映像を同時に画面に表示することができる．

さらに，Logoのリスト処理を用いた人工知能プログラムと３次元タートルグラフィックス機能を合わせて用いることで，タートルの知的な行動を可能にする人工知能アニメーションプログラムを作成することが可能となる．

製品名：3D-LOGO
価格：38,000円
問い合わせ先：ユニー㈱バイナス事業部　03-3448-1060



ロゴライター



3D-LOGO

プログラミング用語
バイナリモード　ファイルに対して制御文字を含む全ての文字列を記憶，読みだしができる．

PROGRAMMING LANGUAGE

特集・この言語を使いなさい!!

日本語プログラミング言語

Mind

(株) リギーコーポレーション 加野島英渡

Mind

私が勧めるこの言語

まずは，欠点から

ではまず欠点からお話しすることにしよう．なんにでも欠点の1つや2つあるのが普通だし常識だから，欠点を隠して長所ばかり見せようとしてもひねくれ……おっと，ものごとを少し斜めから見とおす訓練を普段から積んでいらっしゃる諸兄には聞いてもらえそうもない．

最新のコンパイラと比較すると……

Mindはコンパイラである．インタプリタではなく，ちゃんとした8086のマシンコードを生成するコンパイラである．オブジェクトはそのままCOMファイルになり，MS-DOSの上で動作する．

ただ，ありていにいってしまうがスピードは最新のコンパイラの出すオブジェクトと比較すると少々遅いかもしれない．生成するコードも，どこかの言語のように新人のプログラマよりよっぽど凄いというようなことはけっしてない．そのほとんどはCALL命令と，実におおらかなものだ．

できなくはないが，常駐プログラムを書こうとか，こいつでOSを記述してやろうというような言語ではない．

スモールモデル専用のコンパイラである

コードサイズの上限は64KB，8080モデルではないからデータも別に64KBとれるのだが，これ以上の規模のプログラムを書こうとするとなんらかの工夫が必要になる．できないことはないのだが，言語自体はこうなっているので，そのぶん使うほうでなんとかしてやらなくてはならない．

創意工夫を大いに楽しめる言語だ．

言語の欠点などプログラマの腕次第でなんとでもなる……といったらいい過ぎか.

まだあったかな，そうたくさんはないのだが…….

なぜ私はMindが好きなのか？

さて，私はなぜこういう言語に入れこんでいるのか．それは私がこの言語を作っている会社の社員だから……ではない．私はこの言語に惚れ込んだからこの会社に転職したのだ．それなのに給料が……あ，いやいや，そんなことはどうでもいい．

Mindの特徴はその読みやすさにある．サンプルプログラムを見ていただければ一目瞭然．これがこのままコンパイルされてCOMファイルになる．なにをやっているかすぐにおわかりになるだろう．ついでに私が気付いていないバグまで見つけてしまわれるかもしれない．現にMindプログラマである某氏は，通りすがりにディスプレイを覗きこんだMindの知識のまったくないプログラマにバグを指摘されたという．ありがたいような情けないような話だが，これはほんとうのことだ．

「○○とは　△△して　××することである.」ふむふむ，なるほどね.「なんとか　ならば　どうにかして　次にいろいろする.」ほぉほぉ，わかるじゃないの．まさしく書いてある通りだ．

この，文章を綴るような書きやすさ，読みやすさがMindの身上だし，熱狂的なファンをかかえる由縁でもある．仕様書・即プログラムなどとカッコのいいことをいうつもりはないが，普通どの言語で書いても最上位のレベルになると抽象化が進むので，Mindではこの段階になるとまさに仕様書のようになる．べつにそういう言語をめざしたわけではないのだが，こういう形態の言語なので自然にそうなるのだ．

プログラマの間ではよく「3か月前の自分は他人と同じ」といわれる．自分の書いたプログラムでも3か月もたつと他人の書いたものと同じように読めなくなってしまうというようなことだが，Mindに限ってそのようなことはない．他人の書いたものでもまるで自分で書いたもののようにすらすらと……というわけにはいかないが，ざっと見とおすことならできる．

斜め読みのできるプログラム言語なんてそうそうあるものではない．これがたとえば半角小文字方面のプログラムだとこういうことが可能だろうか．いや，けっしてそういった言語を否定したり非難しようというつもりはない．ネイティブの人たちや，日本人でも英語が得意な人は，それらの言語のソースを日本人がMindのソースをそうするように斜め読みできるのだろうか，という話である．

Mindはどこから来たのか？

MindはFORTHから生まれた……というと，「なんでー，日本語FORTHか」とすぐに早呑み込みする人がいる．そうじゃないんだ，Mindは2つの意味でFORTHから生まれている．

FIFTH86という8086用のFORTHコンパイラがある．これはワード(Cでいう関数のようなもの)の名前に漢字が使えるようになっている．ちょっと乱暴な話だが，それだけではなくこのコンパイラは漢字のワード名に関しては末尾の平仮名をすべて削除して認識するようになっている.「通信ポートをイニシャライズする」を「通信ポートをイニシャライズ」として認識する.「通信ポートをイニシャライズしてよ」でも，ちょっとひねくれて「通信ポートをイニシャライズしない」でもいい（こんな書き方は普通はしないが)．こうすることで，そのシーンに合わせて好きな送り仮名を使うことができるようになるのだ．

さらにFORTH系の言語は，ご承知のように日本語と非常に親和性がいい．printf ("Hello") ;ではなく，"Hello" printなのだ.《表示する(「こんにちは」)》ではなく，《「こんにちは」を　表示する》と表現することができる．FORTH系言語のスタック指向が日本語の語順と実によく合っている．

これをうまく利用すると，初心者にも簡単に使えるような日本語プログラミング言語ができるんじゃないか．書きやすく読みやすい，日本人のためのコンピュータ用言語が作れるんじゃないか．そう考えないほうがどうかしている．このようにMindはFORTH言語のアーキテクチャ，具体的にはスタックの原理を取り入れているという点でFORTHから生まれたということができる．

もちろんMindはFORTHそのものではない．送り仮名も完全に無視するのではなく，特定のシーンでの特定の送り仮名を認識してオブジェクト生成に反映させるといった芸当もする．

たとえば「在籍生徒数から　出席生徒数を　引く」と「出席生徒数を　在籍生徒数から　引く」を同じ結果になるようにコンパイルする「逆転介入」や,「茄子と　胡瓜と　南瓜を　クリアする」といった「併記処理」などが代表的なものである．送り仮名とは無関係だがついでにいってしまうと，この場合茄子がワード型の変数で，胡瓜がダブルワード型の変数で，南瓜が文字

プログラミング用語
バグ　プログラム中の誤動作箇所を差す．米国ではバグから「テントウ虫」を連想し，デバッガのアイコンなどに使われている．

列型の変数であってもいい．「クリア」は配列全体やランダムファイルまでも「クリア」するのだ．ここまでやらないと日本語らしい表現などとても無理である．これをいちいち「ワードクリア」だの「ダブルクリア」だの「ファイルクリア」だのといっていたのでは，とてもじゃないが初心者には使えないし，第一自然じゃない．

MindはFORTHのアーキテクチャから生まれてはいるが，FORTHそのものではない．FORTHは根でMindが花だといったらチャールズ・ムーア氏のお叱りを受けるだろうか．

そしてMindはどこへ向かっているのか

Mindは初心者にも簡単に使えるような言語をめざして開発された．しかしMindはけっして非実用的な言語ではない．PCブリッジ，FM秘書などすでにいくつかの市販のアプリケーションがMindで記述されており，ネットワークではMindで記述されたフリーソフトウェアも多数流通している．

もう少しシステムに近いほうのアプリケーションもある．たとえば製品として売られている機種対応版Mindのパッケージを例にあげると，付属しているMEDITという専用のスクリーンエディタや，Mindシェルというビジュアルシェルはすべてで記述されている．添付のツール類もMindで記述され，一部はソースも付属している．また，大サービスで通信ソフトのソースも付いている．このソースをそのままコンパイルするとひと通りの機能をもった通信ソフトになるので，自分であとは好きなように変えていけばいい．通信，グラフィック，マウスといったライブラリもすべて標準で付属しているので，特別に通信ライブラリなどを買うことはない．これらの機能がすべてサンプルプログラムのように日本語で使えるのだ．たとえば通信のことをいえば，「＜電話番号＞に　接続する」というだけでホストとのリンクが確立する．

このようにライブラリが豊富で使いやすいので，BASICやCで記述されている大概のアプリケーションはMindでもっと楽に作成することができる．もちろん先に書いたような欠点もあるのだが，欠点は技術によって克服できる．また，Mindもいつまでもその欠点をひきずっていくつもりはない．

アプリケーション作成と並んでMindの将来に期待できるのは，学校でのコンピュータ教育用言語としての位置付けである．中学生以上の生徒にコンピュータ言語やプログラムの仕組みを教えるのに，これほど適した言語はほかにない．

プログラミングを教えるということはプログラム言語を教えることではなく，プログラムのロジックを教えることである．別に学生全部に将来プログラマになってほしいとは思わないが，わかりやすく修得しやすいMindでプログラムのロジックを身に付けた学生ならば，将来どの言語を使うようになっても大丈夫だ．

もちろん学校だけではなく，現場でのプログラマ教育にもMindは大いに役だつだろう．無宣言で使えてしまうグローバル変数でひっかかり，メモリにべったりと貼りついたポインタでつまずきしている暇があったら，Mindでもっと大事なことを教えられる．

不思議なことにいちどひとつの言語でプログラミングというものがわかってしまうと，あとはほかのどの言語でも割合楽に移行できる．あ，いや，別に移行を勧めているわけではない．

ちょっとしたサンプルプログラム

ありきたりのものではおもしろくないので，テキストファイル操作のサン

サンプルリスト

```
※タイトル      テキストファイル圧縮ツール（復元不可能（困ったもんだ））
※著作権者      かの(SD100542)
※初版作成      1990/10/16

入力ファイルは          ファイル.

入力ファイル名は        文字列実体.

入力ファイルオープン処理とは    （・　－－＞　・）
        起動パラメータから　単語を切り出して　入力ファイル名に　入れ
        入力ファイル名が　空列？
                ならば　「処理するファイル名を指定してください．」で
                        重大エラー　（DOSに落ちる）
                次に
        入力ファイル名で　入力ファイルを　オープンして　エラー？
                ならば　エラー文字列で　重大エラー
                次に.

範囲？とは                （データ，最小値，最大値　－－＞　真／偽）
                最小値は        変数
                最大値は        変数
                データは        変数
        データと　最小値と　最大値に　入れ
        データが　最小値　以上　かつ
        データが　最大値　以下であること.

平仮名？とは              （データ　－－＞　真／偽）
        'ぁ'から　'ん'までの　範囲？.

メインとは
                処理文字は        変数
                平仮名処理中は　ワード変数
        ブレーク抑止解除して
        入力ファイルオープン処理し
        平仮名処理中を　クリアして
        ここから
                入力ファイルから　一文字読み出して　処理文字に　入れ
                データ終り？
                        ならば　打ち切り
                        次に
                平仮名処理中
                        ならば　処理文字が　平仮名？
                                でなければ
                                        平仮名処理中を　クリアし
                                        処理文字を　一文字表示し
                                次に
                        さもなければ
                                処理文字が　平仮名？
                                ならば　'　'を　一文字表示して
                                        平仮名処理中を　セットして
                                さもなければ
                                        処理文字を　一文字表示し
                                次に
                        次に
        繰り返して
        入力ファイルを　クローズすること.
```

プログラミング用語
ヒープ　ヒープは伸び縮みする動的データ領域でメモリ上にはスタック領域の後ろに確保されます．

プルをお目にかけたい（なんだ，ありきたりの権化のようなテーマじゃないか）．

テキストファイルの中から平仮名だけを削除するプログラムで，実用性はまったくない．ただ，表示されたものを眺めているとけっこう笑えるから，Mindの処理系をおもちの方はぜひお試しあれ．一部では「中国語フィルタ」などと呼ばれているようだが，もちろんそんな大層なものではない．平仮名がなくてもけっこう意味が通じてしまうところが日本語の凄いところで，Mindもこれほど乱暴ではないがこんなふうにして単語を認識している．

終りに

ところでさきほど，Mindは２つの意味でFORTHから生まれていると書いた．１つはすでに説明したが，もう１つの意味は，Mindのコンパイラ自体はその発想の母体になったFORTH系コンパイラであるFIFTH 86で記述されているということだ．はたしてMindからはなにが生まれてくるのだろうか．

あ，そうそう，３つめの欠点をいま思い出した．わかりやすく読めてしまうので，神秘性やありがたみがない．

これはかなり大きな欠点かもしれないなぁ．

処理系紹介

Mind

Mind 5.1

現在Mindのコンパイラには，機種依存版，MS-DOS汎用版，フリーソフトウェア版の３種類が存在している．

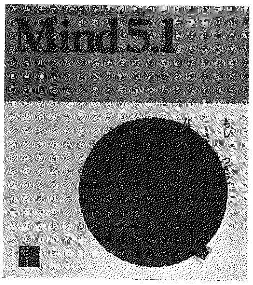
●機種依存版

機種依存版としては，(株)マイクロソフトウェアアソシエイツからPC-9801/PC-286対応版が，富士通(株)からFMRシリーズ対応版が発売されている．

機種依存版は，エディタ，コンパイラ，オンラインヘルプなどを統合して開発が行えるビジュアルシェルと機種に依存したグラフィックス，通信などのライブラリを標準装備している．

●MS-DOS汎用版

MS-DOS汎用版はPure Mind Pro 5.1と呼ばれ，バージョン2.11以上のすべての日本語MS-DOSの上で動作する．



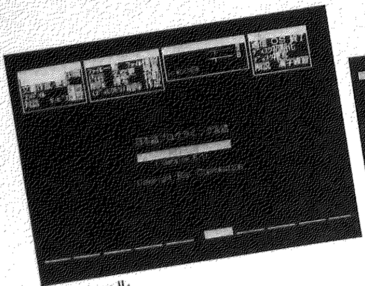
製品名：Mind 5.1
価　格：39,800円
製品名：Pure Mind Pro 5.1
価　格：28,000円
製品名：デバイスドライバ＆常駐プログラム作成キット
価　格：10,000円
問い合わせ先：㈱マイクロソフトウェアアソシエイツ
03-3476-1411

Pure Mind Proには統合環境や機種依存したライブラリは付属していないものの，すべてのランタイムライブラリのソースコードが付属している．また，Pure Mind Proとデバイスドライバ＆常駐プログラム作成キットと組み合わせることで，簡単にデバイスドライバや常駐プログラムを作成することが可能である．

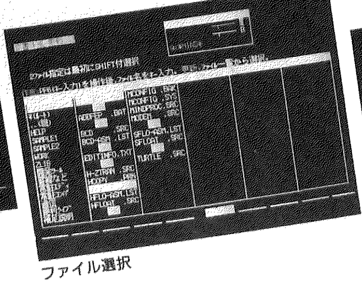
●フリーソフトウェア版

これは，Mindのユーザー会であるMindClubによって主要なBBSなどを通じて配布されているもので，パブリックMindと呼ばれている．ユーザーコード領域が10KBに制限されているが，DOS上のツール作成などに十分実用になる．

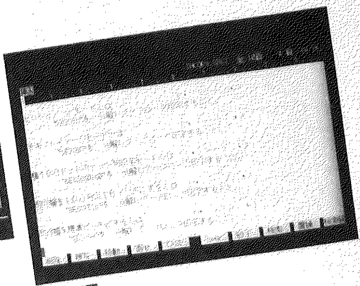
上記以外にも，富士通の日本語ワープロ専用機OASYS30AX用の日本語MS-DOSにはMindがバンドルされている．



Mindシェル



ファイル選択



MEDIT

プログラミング用語
引数　関数や手続きの呼び出し時，引数で値を渡す．

モジュール化を重視した文法

Modula-2

西端芳彦

なんでCとBASICなの？

パソコンでプログラムを書くといえば，出てくるのがCとBASIC. さすがに最近は何でもBASICという人は減ったみたいだけど，そのぶんCの人気が高いみたいで，みんなCを使っている．日本人って右へならえの体質があって，Cが流行りはじめるとC一色になってしまったわけだけど，言語にはそれぞれ適材適所というものがあるわけで，Cしか知らないから何でもCというのは不毛なのではないかなと思うわけ．

そんなあなたにModula-2

LispとかPrologを使うべき人っていうのは自分で判断できるよね？　でもってそうじゃない人は，いわゆる手続き型の言語を使うはず（あ，Object Orientedの話はここでは置いとくことにする．SmalltalkやC++の記事を読んでね）．

そこで，いまCを使っていて何の不自由も感じていない人，それからUNIXマシンを使っている人，こういう人達は迷わずCを使っていてください．

というわけで，残った人はModula-2を使うと幸せになれる（かもしれない）というのがとりあえずの結論．

なぜModula-2なの（その1、歴史的側面）

昔むかしスイスにNiklaus Wirthという偉い先生がいた（ほんとはいまでもいるのだ）．Wirth先生は，ある日

プログラミング用語
標準エラー出力　標準入力エラー出力ストリームの結合先、ディフォルトではディスプレイが指定されている。

Pascalという言語を作った．Wirth先生は自分の教科書をPascalで書くとかコンパイラをみんなにあげるとかしたので，Pascalは世界に広まっていったのだった．

ところが，Wirth先生がPascalを作った動機というのが学生にプログラムを教えるためだったので，実際の仕事に使おうとするといろいろ機能の足りないところが出てきた．そういうときには普通なら機能を追加した新しいPascalを作るでしょ？　だけど，Wirth先生はまた新しい言語を作ってしまった．そう，こうしてModula-2は誕生したのだ．

いっぽうCは，UNIXの記述のために作成された．UNIXを開発したのはご存知の通りベル研のバリバリのプログラマ達だ．Cはもともとそういう連中が使いやすいように作られている．裏を返せば，もともとは素人さんお断りの言語だった．ちなみに，僕もその昔（日本にUNIXが入ってきたころね）友人にC教えてっていったら，あんたらの使う言語じゃないって断られてしまったのだ．

こうした出自の違いがModula-2（やPascal）とCとの性格の違いに結びつくわけで，すぐにミスをする学生さんを念頭に置いて作ったPascalやその後継者というべきModula-2が，プログラマのミスをコンパイラが見つけやすいようになっているのに対して，Cは違反すれすれの技に溺れたプログラミングも可能になっているわけだ．

だから，ケアレスミスが多いと感じている人，Cでプログラムを書いていてやたらと暴走させている人，これからプログラムをはじめてみようという人には，Modula-2をお勧めしちゃいましょう．

なぜModula-2なの（その2、Pascalとの相違）

それならPascalでもいいんじゃないって声も聞こえてきそう．パソコンでPascalを使うなら，MS-DOS用にはTurbo Pascal，Macintosh用にはTHINK Pascalっていうとってもいい処理系があるんだものね．

Modula-2はWirth先生がPascalじゃ能力が足りないというので新たに作った言語なのだけど，だからといってTurbo PascalやTHINK Pascalよりすぐれものという風に簡単にはいかないのだ．なぜって，TurboやTHINKも標準Pascalでは足りない機能をいろいろ拡張してあるからね．

さて，Modula-2はPascalとどこが違うかというと，構文が少し変わった，なんていうのは置いといて．

システム寄りの機能が入った

システム寄りの機能としては，Cのキャストに相当するtype transferとか，アドレスが入るADDRESS型とかがある．

とはいうものの，こういった機能はシステムプログラミングには不可欠なので，Turboをはじめ実用に供されるPascalの処理系にも当然用意されている．

MODULEによる分割コンパイルが可能になった

Wirth先生が設計した段階では，Pascalには分割コンパイルの機能がなく，ソースプログラムは1つのファイルにすべて入るようになっていたのだけれど，これではライブラリが使えないとかいろいろ不便なことが多い．

で，分割コンパイルできるようにしたのだけど，これはModula-2が最初ではなくて，カリフォルニア大学サンディエゴ校で作成されたUCSD Pascalのほうが早かったのだ，実は．

で，TurboもTHINKもこのUCSD流の分割コンパイルを採用しているので，問題はない．ただ，Modula-2はWirthがXEROXのPARCに滞在したときに見たとかいうMESAの影響を受けて，プログラムのモジュール化をより重視した文法になっている．

モジュール化というとなんだか難しそうだけど，何をしたいかというとサブルーチンをうまく整理して，別のプログラムでも利用できるようにしておきたいってこと．

で，Modula-2の分割コンパイルは

リスト1

```
ファイル1：complex定義モジュール

DEFINITION MODULE complex;

(* データ型の定義 *)

TYPE complex = RECORD
        realPart,imaginaryPart : REAL;
END;

(* 他のモジュールで使用される手続き *)

PROCEDURE MakeComplex(r,i: REAL) : complex;

PROCEDURE AbsComplex(x: complex) : REAL;

PROCEDURE WriteComplex(x: complex; w,n: CARDINAL);

END complex.

ファイル2：complex実現モジュール

IMPLEMENTATION MODULE complex;

(* システムで用意されたモジュールからのインポート *)
```


プログラミング用語
標準入出力　標準入出力ストリームの結合先．ディフォルトではディスプレイとキーボードが指定されている．

どうなっているかというと，リスト１のようにメインプログラムのあるファイル以外は定義モジュールと実現モジュールの２つのファイルに分かれている．通常のソースファイルに相当するのは実現モジュールで，定義モジュールのほうにはよそのファイルから利用するときのための情報だけが入っているわけ．

ちょっと考えると１つのファイルになってたほうがよさそうだけど，実は分れているほうが便利．プログラムを修正したときでも，定義モジュールに変化がなければほかのファイルは修正の必要がないって機械的にわかるからね．とくに最近は２つ以上のファイルが同時に編集できるエディタなんてあたりまえだから，ファイルが分かれていても不自由はないし．

それに定義モジュール自体がそのモジュールの説明になるからね．各手続きの仕様までコメントで入れておけば，使うときには不自由しないはず．

それからModula-2の名前（変数名や手続き名なんか)は，IMPORTを使ってどのモジュールで定義した名前かを指定するようになっている．だからモジュール名だけちゃんと決めておけば名前が衝突することがない．

ひとりでプログラム書いている分にはあんまり関係ないけれど，大勢で協力して１つのプログラム書くときにはとっても役に立つ機能だ．これがないと手続き名やグローバル変数名まで，名前を付けるルールを決めておかないと最後にリンクのところで怒られてしまう．

というわけで，Modula-2は大規模なプログラム開発に向いているはずなんだけど，なぜかあんまり流行らない．不思議，不思議．

```
FROM MathLib IMPORT Sqrt;
FROM InOut IMPORT WriteReal,WriteString;

(* 手続きの本体 *)

(* 新しい複素数型の変数をつくる *)

PROCEDURE MakeComplex(r,i: REAL) : complex;
        VAR
                c : complex;
BEGIN
        c.realPart := r;
        c.imaginaryPart := i;
        RETURN c;
END MakeComplex;

(* 複素数の絶対値を計算する *)

PROCEDURE AbsComplex(x: complex) : REAL;
        VAR
                value : REAL;
BEGIN
        WITH x DO
                value := Sqrt(realPart * realPart + imaginaryPart * imaginaryPart);
        END;
        RETURN value;
END AbsComplex;

(* 複素数を表示する *)

PROCEDURE WriteComplex(x: complex; w,n: CARDINAL);
BEGIN
        WITH x DO
                WriteReal(realPart,w,n);
                WriteString(" + ");
                WriteReal(imaginaryPart,w,n);
                WriteString("i");
        END;
END WriteComplex;

END complex.
ファイル3 : メインプログラム

MODULE main1;

(* メインプログラム *)

(* システムで用意されたモジュールからのインポート *)

        FROM InOut IMPORT WriteLn,WriteReal;
        FROM complex IMPORT complex,MakeComplex,AbsComplex,WriteComplex;

VAR
        c : complex;
        a : REAL;
BEGIN
        c := MakeComplex(1.0,1.0);
        a := AbsComplex(c);
        WriteComplex(c,10,5);
        WriteLn;
        WriteReal(a,10,5);
        WriteLn;
END main1.
```

リスト２

```
ファイル2 : complex定義モジュール

DEFINITION MODULE complex;

TYPE complex; (* データ型は型名だけ *)

(* 他のモジュールで使用される手続き *)

PROCEDURE MakeComplex(r,i: REAL) : complex;

PROCEDURE AbsComplex(x: complex) : REAL;

PROCEDURE WriteComplex(x: complex; x,w: CARDINAL);

END complex.

ファイル2 : complex実現モジュール

IMPLEMENTATION MODULE complex;

(* システムで用意されたモジュールからのインポート *)

FROM Storage IMPORT ALLOCATE;
FROM MathLib IMPORT Sqrt;
FROM InOut IMPORT WriteReal,WriteString;
```

プログラミング用語
標準補助入出力　標準補助入出力ストリームの結合先．ディフォルトではRS232ポートが指定されている．

では例題を…

などと説明してみても，具体的なイメージがつかめないと思うので例題を少し．

まずはさっきのリスト1．これは複素数を扱うライブラリの例だけど，ごく普通のプログラムだ．CやPascalで書いてもほとんど同じなんじゃないかな．出力のライブラリは違うけど...

次にリスト2．今度はデータ型を隠蔽する機能を使って書いてみた．もちろん複素数を扱うだけならこんなよけいなことはしなくていいのだけれど，例として使ってみただけ．ほんとはもっと複雑なデータ型の場合に使えばいいのだけど...

それはともかく，今度は定義モジュールにcomplexという型の定義が入っていない．単に型の名前があるだけだ．で実現モジュールのほうに型の定義が入っている．ここでcomplex型がポインタになっているのは，隠蔽型はポインタでないといけないというModula-2の制限によるものだ．この例の場合は，とくにポインタを使う理由はほかにない．

で，このように定義モジュールに型の定義がないということは，よそのモジュールからはcomplex型の中身はわからないということで，裏を返せばプログラムの都合で定義を途中で変更しても，このモジュール以外の部分を修正する必要がないことになる．何万行にもなるような巨大なプログラムでは，あとからデータ型を変更するとなるとたいへんなことになるが，このように隠蔽型を使用すればモジュール内の修正だけで済む．いわゆる抽象データ型(abstruct data type)というやつだ．

```
(* データ型の定義 *)

TYPE complexData = RECORD
                    realPart,imaginaryPart: REAL;
        END;
         complex = POINTER TO complexData;

(* 手続きの本体  *)

(* 新しい複素数型の変数をつくる *)

PROCEDURE MakeComplex(r,i: REAL) : complex;
        VAR
                c : complex;
BEGIN
        ALLOCATE(c,SIZE(complexData));
        c^.realPart := r;
        c^.imaginaryPart := i;
        RETURN c;
END MakeComplex;

(* 複素数の絶対値を計算する *)

PROCEDURE AbsComplex(x: complex) : REAL;
        VAR
                value : REAL;
BEGIN
        WITH x^ DO
                value := Sqrt(realPart * realPart + imaginaryPart * imaginaryPart);
        END;
        RETURN value;
END AbsComplex;

(* 複素数を表示する *)

(* システムで用意されたモジュールからのインポート *)

PROCEDURE WriteComplex(x: complex; w,n: CARDINAL);
BEGIN
        WITH x^ DO
                WriteReal(realPart,w,n);
                WriteString(" + ");
                WriteReal(imaginaryPart,w,n);
                WriteString("i");
        END;
END WriteComplex;

END complex.

ファイル3：メインプログラム

MODULE main2;

(* メインプログラム *)

(* システムで用意されたモジュールからのインポート *)

        FROM InOut IMPORT WriteLn,WriteReal;
        FROM complex IMPORT complex,MakeComplex,AbsComplex,WriteComplex;

VAR
        c : complex;
        a : REAL;
BEGIN
        c := MakeComplex(1.0,1.0);
        a := AbsComplex(c);
        WriteComplex(c,10,5);
        WriteLn;
        WriteReal(a,10,5);
        WriteLn;
END main1.
```

そして，いろいろ……

Modula-2の機能が簡潔にまとまっているといっても，このページ数じゃ全部の紹介はできるわけない．Modula-2の大きな特徴で，この記事で取り上げなかったのはコルーチン．ぼくもよくわかってないという切実な理由もあるのだけれど，Modula-2のすべての機能や概念を羅列するよりは，単にPascalやCのかわりに使うにしても，こんなにいいんだよって話をしたかったからなのだ．もっと内容のある記事を期待してた人ごめんなさい．

パソコンで動く処理系の紹介は別にあると思うけど，Modula-2の処理系ってわりと安く（機能にしてはね）手に入るので，興味のある人は実際に動かしてみてください．言語は実際に使ってみなくちゃね．

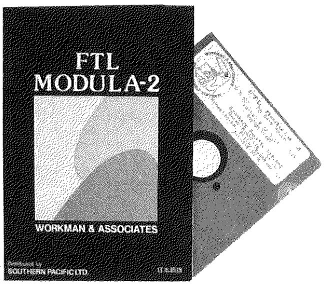
プログラミング用語
ブール演算　オペランドおよび演算結果が2値からのみなる演算．

処理系紹介

Modula-2

FTL MODULA-2

米WORKMAN&ASSOCIATES社のModula-2コンパイラ．Modula-2の開発者N.Wirth教授によるModula-2第3版に準拠している．エディタ，コンパイラ，リンカ，メイクなどの各種ユーティリティを統合環境で利用できること，5秒間に1000行のソースプログラムをコンパイルする高速性が特長．8087ライブラリが付属し，割り込みベクタトラップライブラリでポップアップアプリケーションの作成が可能．NEC PC-9800シリーズで動作可能．ほかにCP/M版もある．



製品名：FTL MODULA-2
MS-DOS版
価格：19,800円
問い合わせ先：㈱サザンパシフィック
045—314—9514

LOGITECH MODULA-2 Ver.3.0

米ロジテック社開発のModula-2コンパイラ．Modula-2第3版に準拠している．製品の形態には，Modula-2の基本的な開発環境を構築するためのコンパイラパック，各種開発ツール類を集めたツールキット，この2つがセットになったデベロップメントシステムとPascalのソースプログラムをModula-2プログラムに変換するトランスレーターがある．NEC PC-9800シリーズ、東芝J3100シリーズ、富士通FMRシリーズ、各社AXパソコンに対応。

●コンパイラパック

コンパイラ，専用リンカ，エディタ，ライブラリ，ライブラリソース，プログラムの異常終了時の状態を調べるデバッガが含まれている．

●ツールキット

ランタイムデバッガ，デコーダ，メイクユーティリティ，クロスリファレンサ，ソースファイルチェッカ，ソースマネージャ，ソースフォーマッタが含まれる．

●デベロップメントシステム

コンパイラパック＋ツールキットのセット．

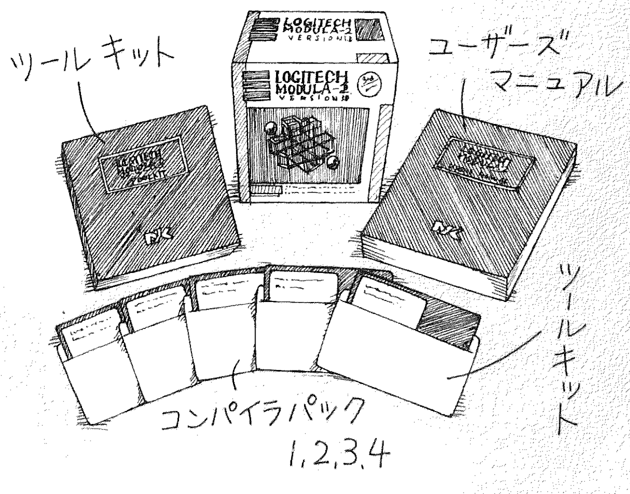
●Pascal to Modula-2トランスレーター

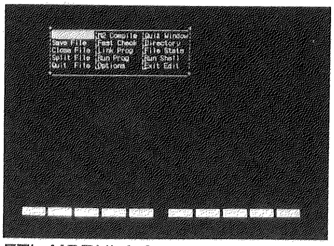
標準PascalやTurbo Pascal Ver.3.0のソースプログラムをModula-2プログラムに変換する．

●NEW ATOM

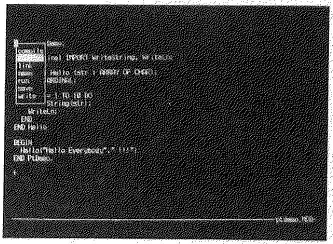
NEC PC-9800シリーズ専用のグラフィックライブラリ．

製品名：コンパイラパック
価格：39,000円
製品名：ツールキット
価格：29,000円
製品名：デベロップメントシステム
(コンパイラパック＋ツールキット)
価格：59,800円
製品名：Pascal to Modula-2
トランスレーター
価格：9,800円
製品名：NEW ATOM
価格：19,800円
問い合わせ先：㈱NJK
03-3499-2730

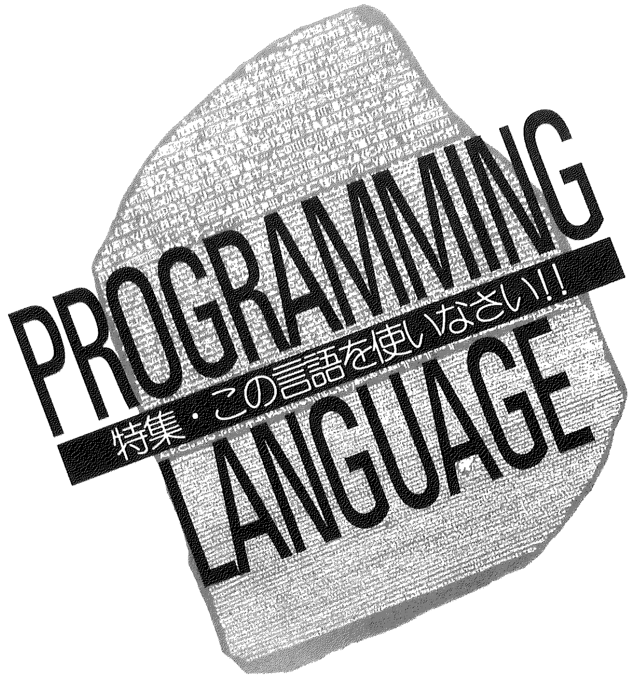




FTL MODULA-2



LOGITECH MODULA-2 Ver.3.0



ミスを適切に指摘してくれる優しさ

Pascal　オケゲム

なぜ，Pascalが好きなのか？

Pascalには大向う受けをねらうような機能は１つもありません．けれども，私はPascalが大好きです．それは，Pascalは優しい言語だからです．私がPascalを好きだというのいちばんの理由はここにあるので，まずPascalの優しさについて説明しましょう．

私は趣味でコンパイラのおもちゃを書いて遊んでいます．以前はＣ言語で記述していたのですが，パソコン上でも本格的なPascalコンパイラ（TurboPascal Ver. 5. 5)が出てきたので，おもちゃコンパイラをPascalに移植してみました．

Pascalに移植してみると，開発・保守に対する負担がすごく軽くなったのに気づきました．この理由はいろいろありそうですし，個々の理由についてはあとのほうで説明したいと思います．ただ，Pascalには，本質的なところで「ユーザーはまちがいを起こしがちな生き物である」というポリシーがあるような気がするのです．これに対して，Pascalとよく対比されるＣ言語には，「ユーザーの指示したことには絶対服従．ユーザーはまちがいを起こさない」という前提があるような気がしてなりません．

いうまでもなく人間はしばしばミスを犯しますが，Pascalは人間のミスに寛容です．寛容というのはミスを大目に見るということではなくて，ミスを適切に指摘してくれるという意味です．このへんが，Pascalが優しく感じられるゆえんかもしれません．

プログラミング用語
プロトタイプ　関数定義時に行う，引数の型宣言のこと．ANSI Cではこのプロトタイプ宣言で引数の型をチェックする．

どうしてPascalが生まれたのか

Pascalはスイスの N. Wirthが1971年に開発した言語で，元々は教育用として設計されました．

この当時は，第3世代のコンピュータが登場してハードウェアが大いに普及したのに対してソフトウェアの供給が追いつかないという「ソフトウェア危機」が騒がれだした頃にあたります．この危機を解決するための方法論，すなわち構造化プログラミングを教育するために，Wirthは次のような目標をPascalに与えました．

(1)プログラミングを，種々の基本概念の組合せとして系統的に訓練できること．

(2)既存のコンピュータ上で効率よく信頼性の高いコンパイラを開発できること．

これらの要求を，Wirthがどのようにして実現したかを紹介していくことにします．

言語仕様上の特徴

N. Wirthは，コンピュータプログラミングを2つの本質的部分の記述からなると考えました．それは，動作 (algorithm) の記述とデータの記述です．このため，単純で系統だった方法でデータとアルゴリズムを記述できることがPascalの大きな特徴となっています．

これらのうちアルゴリズムの記述，すなわち制御構造についてはAlgol60から受け継いだものでありC言語とも同等の機能なので，ここではPascalのもっとも大きな特徴であるデータ構造の記述能力のほうを紹介することにします．

データ構造とは現実世界の抽象表現と考えることができます．御存知の通り，コンピュータの内部では，データは0，1のビット列として表現されていますが，Pascalではビットレベルの実現方法を意識しないでプログラムすることが可能です．ビット列よりも人間にとってわかりやすい表現を提供しているからです．

たとえば，Pascalには列挙型というものがあります（図1）．

意味するところは一目瞭然でしょう．これはほんの1例ですが，Pascalで記述されたプログラムはたいへん読みやすいので，ソースリスト自体がドキュメントの働きをもっていることも特徴の1つです．

Pascalにはこの列挙型の外に，文字型，数値型，部分範囲など基本となる多くの型があります．また，利用者がこれらの基本型をまとめて複雑な型を作ることもでき，その手段としてレコード，配列，集合，ポインタ，ファイルなどの構造型が用意されています．

これら豊富な型機能を利用して，人間に取ってわかりやすい抽象表現によるプログラミングが可能になるわけです．

しかし，抽象表現によるプログラムは効率が悪くならないでしょうか．じつはそうではなく，Pascalは現実のCPUでの効率にも配慮されています．なぜなら，さきほど紹介した型の機能は，現存のCPUの機能をモデル化したものでもあるからです（表1）．

図1．列挙型

```
colors = (red, yellow, green);
```

表1　型の機能とCPUの機能

型	CPU上の表現
列挙型	数値（ビット列）
配列	インデックス修飾
レコード	インデックス修飾
ポインタ	間接参照

表1のように，Pascalでは現実世界の抽象表現とコンピュータ内部での具体的表現の両立という矛盾する条件を絶妙なバランスで実現しています．

ここで，Pascalの特徴を復習してみることにしましょう．今まで出てきたPascalの特徴は次のようにまとめることができます．

- 構造的な制御構造がある
- 複雑なデータ型を作ることができる
- チェックが厳密である
- 効率がよい
- プログラムが読みやすい

これらの特徴は，教育のみならず実務で大規模なプログラムを作成する際にも必要なものです．

とくに言語をプロの道具として見るとき，プログラムを最初に開発するときの書きやすさよりも，保守していくときのわかりやすさ，安全性が非常に重要です．プログラムが読みやすく，チェックが厳しいというPascalの特性は，メンテナンスに多くの時間を費やしている現場のプログラマにとっては福音となります．

このため，Pascalは教育だけでなくアプリケーション開発にも広く利用されています．有名なところでは，Cray社のスーパーコンピュータ用最適化FORTRANコンパイラや，Knuthが開発した文書処理系のT_EXがPascalで記述されています．

最近ではMS-DOS上のPascalコン

プログラミング用語
分割コンパイル　1つのプログラムを複数のソースファイルで作成し，別々にコンパイルした後，実行ファイル作成時リンクすること．

サンプルリスト

```
program printList(input, output);
type
    alpha = string[32];
    refNode = ^node;
    node  = record
                name : alpha;
                next : refNode
            end;
var
    root : refNode;

function getName(var nm : alpha) : boolean;
begin
    write('>'); readln(nm);
    getName := nm <> ''
end;

procedure makeList;
var
    p : refNode; nm : alpha;
begin
    while getName(nm) do begin
        new(p);
        p^.name := nm;
        p^.next := root;
        root := p
    end
end;

procedure print(p : refNode);
begin
    if p <> nil then begin
        print(p^.next); { 再帰呼出 }
        writeln(p^.name)
    end
end;

begin               { メイン }
    root := nil;    { リストの初期化 }
    makeList;       { リストの作成 }
    print(root)     { リストの表示 }
end.
```

パイラも整備され,ライブラリも豊富になってきました.Pascalはアプリケーションソフトを開発するのに最適な言語といえそうです.

具体的なプログラミング

Pascalによる簡単なプログラム例として,ポインタによる線形リストの作成と表示のプログラムを紹介します.

限られた紙面では,おもちゃのプログラムしか載せられませんが,Pascalの特徴であるデータ構造の記述法だけは削りたくなかったので,ちょっと人工的な例題になってしまいました.

プログラムは,プロンプトに対して文字列を入力していき,最後に空行を入力すると,それまで入力した文字列を順に表示するというものです.

データ上は入力と逆順に並んでいますが,出力の際に再帰呼出しを利用してデータの逆順に出力するので,結果として入力と同じ順番に出力されます.ちょっとひねくれたプログラムです.

FORTH

1960年代後半に,Charles H. Mooreによって,仮想スタックマシン上で動くアセンブリ言語として発表され,1974年に完成した.一般に言語は研究室で開発されるのに対し,FORTHはひとりの技術者によって開発された言語であった.インタプリタの柔軟性をもちながら実行速度はコンパイラに近く,また拡張性にも非常に高い.

FORTHはスタックマシンのアセンブリ言語であるため,数式表現は「3+5」ではなく「3 5 +」と逆ポーランド表記法をもちいる.この構文は英語よりも日本語に近いものとなり,日本人にとってきわめて読みやすく書きやすい言語となっている.

FORTHは一般にコンパイラに分類されるが,かなり性格を異にする,またインタプリタとも違う.FORTHは機械語のサブルーチンの組み合せを「ワード」という単位で管理,実行をおこなう.FORTHのコンパイルとは,古いワードを組み合せたり,機械語のサブルーチンを記述して,インタプリタにそれぞれのアドレスを新しいワードとして登録することである.実行とは,アプリケーション内部の数十バイトのインタプリタに登録されている各ワードのアドレスを順に実行していく.ワード自体は大きくても数十バイトの機械語サブルーチンの集まりであるが,各ワードを実行する度たびにインタプリタが呼び出される.この単純さが大規模なアプリケーションの実行時に速度の低下をまねいている.

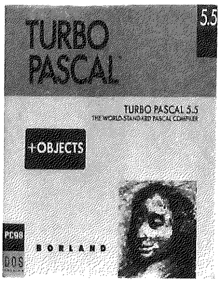
プログラミング用語
ヘッダファイル　ソースプログラムに関連した関数の集合,定数の集合宣言などヘッダ部分を独立したファイル.

処理系紹介

Pascal

Turbo Pascal

米Borland社開発の統合環境型のPascalコンパイラ．Ver.5.5ではオブジェクト指向プログラミングを取り入れ，object，constructor，destructor，virtualの4つの予約語を追加してC++風の機能を実現している．また，新たにユニットという概念を導入して分割コンパイルを可能にしている．



製品名：Turbo Pascal 5.5
価格：29,800円
製品名：Turbo Pascal 5.5 Professional 2nd EDITION
価格：49,800円
問い合わせ先：㈱ボーランドジャパン
03-3486-1400

Pro PASCAL

Prospero Software社のPascalコンパイラ．ISO7185スタンダードに正確に準拠し，大型機やミニコン上のPascal処理系との互換性が高い．スモール，エクステンデッド，ジャンボの3種類のメモリモデルが指定可能．シンボリックデバッガPROBE，高速オーバーレイリンカPro-LINK，多機能ライブラリアンPro-Libを含む．

NEC PC-9800シリーズ，富士通FMR-50/60/70およびFM TOWNS，松下Panacom M500/700/800，IBM PS55/5550/5560/5570，PC/AT，東芝J-3100/3300シリーズに対応．

製品名：Pro PASCAL
価格：98,000円
問い合わせ先：(株) ライフボート
03-3293-4711

Advanced RUN/PASCAL

Pro PASCALのサブセット版コンパイラ，専用リンカ，日本語フルスクリーンエディタ，対話型シンボリックデバッガPROBEのセット．プログラミング言語の学習環境を提供することを目的としたライフボート社のAdvanced RUNシリーズのひとつ．

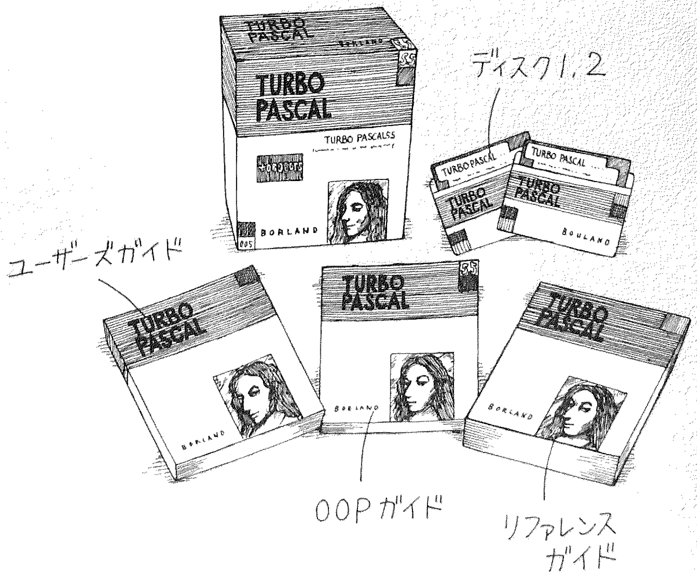
インターフェースはBASICライクなコマンドベース．Pro PASCALとの互換性があるため開発したソースプログラムはそのままPro PASCALで使用可能．

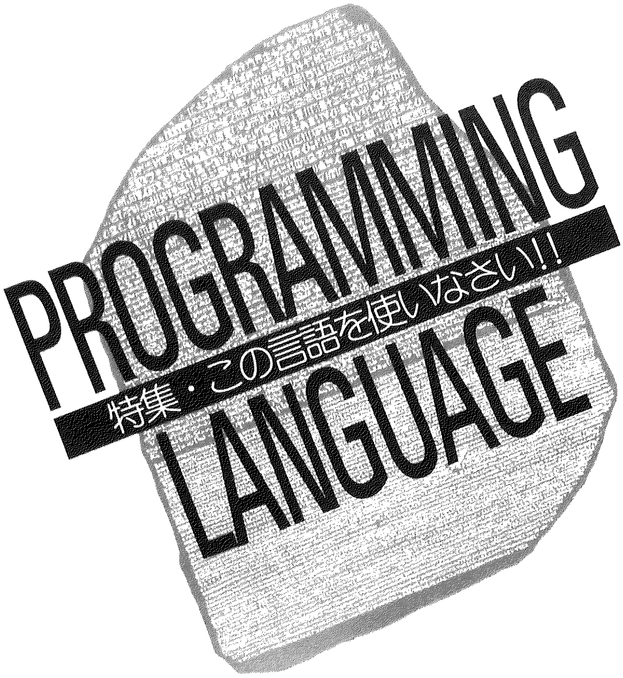
NEC PC-9801シリーズ，富士通FMR50/60/70およびFM TOWNS，IBM PSシリーズ，東芝J-3100シリーズに対応．

製品名：Advanced RUN/PASCAL
価格：29,800円
問い合わせ先：(株) ライフボート
03-3293-4711

Turbo Pascal

Advanced RUN/PASCAL

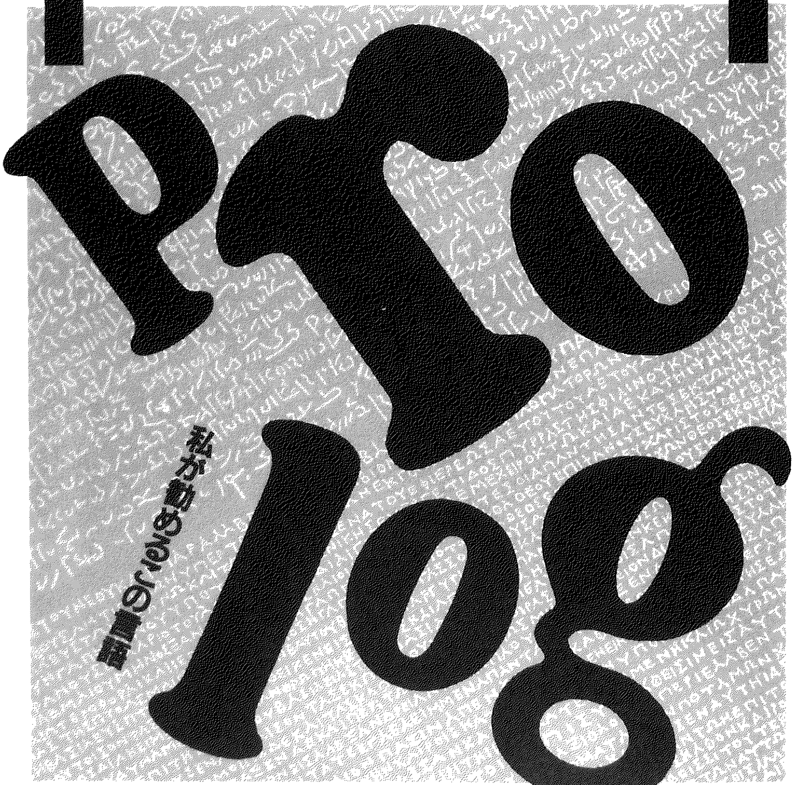




強力なプログラムを速く記述する

Prolog

鈴木太朗



プロローグ

Prologという名は，PROgramming in LOGicに由来しています．その名のとおり，論理（述語論理）を基礎にしたプログラミング言語です．CやFORTRANなどの言語はコンピュータの動作原理を忠実に反映した言語仕様をもっています．

それに対して，Prologは「論理」という数学的なモデルを反映しています．このため，Prologはほかの言語に比べて人間向きであるといえます．Prologのように論理を用いてプログラミングを行うことを「論理プログラミング」といいます．

記号論理学の応用から

19世紀後半に発達した記号論理学は「人間の推論過程の機械化」を目的としていました．これをプログラミングに応用することで，高度な推論処理をコンピュータ上で行いたいと考える人たちがいました．1965年に，ロビンソンはコンピュータ向きの推論法（導出原理）を発表しました．また，コワルスキーは導出原理を計算の過程とみなすことで，論理式の集合をプログラムとみなすことができるということを証明しました．これと前後して，1970年代前半にコルメイアが最初のPrologシステムを開発しました．

これ以後，Prologに代表される論理プログラミングはヨーロッパで研究，開発が進められていきました．

1979年にウォーレンがDec-10上で効率の良い処理系を実現したことで，論理プログラミングは十分実用になる

プログラミング用語
ポインタ　間接的に各種変数を参照するオブジェクトのこと．文字変数などに利用され，変数自体はその格納アドレスを指す．

ことが実証されました．その後80年代になって，日本のICOTが論理プログラミングを核とした第５世代コンピュータ計画を発表するに至って，論理プログラミングは世界中で研究されるようになりました．

ユニフィケーションによる推論

Prologの言語としての特徴としては以下のものがあげられます．

・宣言的なプログラミング
・ユニフィケーションによるパターンマッチ機能
・バックトラックによる探索機能

これから，簡単なプログラミングを通してこれらの特徴について具体的にみていきます．

かえるは泳げる

```
can_swim(frog).
```

たった１行だけですが，これでもりっぱ（？）なPrologプログラムです．上のプログラムは１つの「節」からできています．節は「.」で区切られます．can_swim は「述語」といいます．かっこの中にあるものは「引数」です．述語は引数の性質，もしくは引数同士の関係を表します．小文字で始まる引数は「定数」といい，ある特定の対象を表します．このプログラムは「かえる（frog）は泳げる（can_swim）」という「事実」を表しています．Prologプログラムは論理式とみなせるので，プログラマが述語名や定数名を適切に選びさえすれば，その意味を他人に理解させることは容易です．このように，実行に関わらず字面だけでその意味を把握できるようなプログラミングを「宣言的なプログラミング」といいます．

かえるは泳げるか？

それでは，このプログラムを実行してみましょう．実行は，Prologのシステムに「質問」をすることで始まります．つぎのような質問をしてみます．

```
?- can_swim(frog).
```

これは「かえるは泳げるか？」という質問を表します．これを入力すると

```
yes
```

が返ります．これは，質問のcan_swim（frog）とプログラムのcan_swim（frog）がユニフィケーションにより，マッチしたためです．ユニフィケーションとは論理学の用語で「統合」とか「統一化」という意味です．これは，パターンマッチをより強力にしたものと思っていただければけっこうです．

こんどは次のように質問してみます．

```
?- can_swim(X).
```

これは「泳げるものは何か？」という質問を表しています．上の質問中のXは「変数」です．これを入力すると

```
X = frog
```

と返ってきます．これは，質問のcan swim（X）とプログラムのcan_swim（frog）のユニフィケーションによって，変数Xが定数frogとマッチしたためです．

こいは泳げるか？

次に前の１行プログラムを少し拡張します．

```
can_swim(frog).
can_swim(X)  :- fish(X).
fish(carp).
```

３行目は「こい（carp）は魚（fish）である」という事実を表しています．では，２行目は何でしょうか？これは「もしXが魚ならば，Xは泳げる」という「規則」を表しています．「規則」は一般的には以下のような形をしています．

```
A :- B1,B2, ... ,Bn.
```

これは，「もし，B1かつB2かつ ... Bnならば，Aである」という意味を表します．

このプログラムに「こいは泳げるか？」と質問をしてみます．

```
?-can_swim (carp).
```

これを実行すると，まずプログラムの１行目can_swim（frog）とのユニフィケーションを試みます．しかし，これはcarpとfrogがマッチしないので失敗します．そこでシステムは１行目とのユニフィケーションを御破算にして２行目とのユニフィケーションを試みます．このようにある時点まで後戻りして，ほかの可能な処理を行うことを「バックトラック」といいます．これは，探索問題などに対して威力を発揮します．

プログラムの２行目とのユニフィケーションは，質問中のゴールcan_swim（carp）と:- の左辺can_swim（X）とに対して試みられます．これは成功し，規則に含まれる変数Xはcarpとマッチします．:-の両側に現れるXは同じものを表しているので，:-の右辺はfish（carp）になります．

:-の左辺とのユニフィケーションに成功すると，システムは:-の右辺を新たな質問とみなして処理を続行します．したがって，今度はfish（carp）とマッチするものを探します．これは３行目と一致するので，結局

```
yes
```

が返ります．このように規則を使うことで，新たな事実をプログラムから引き出すことができます．

プログラミング用語
前判定反復文　制御文の一種で，while〜文のように継続条件を前で判断する文．

Prologプログラミングの利点

このようなプログラミングをすることで，プログラマはどのような恩恵を受けられるのでしょうか？　ひとつは，プログラムを読みやすくできるということです．これにより，デバッグや保守が容易になります．また，論理的に解釈できることを利用して，プログラムの正しさを静的に検証したり，既存のプログラムから新たなプログラムを合成することが可能です．

ふたつめは，強力なプログラムを速く書けるということです．ユニフィケーションやバックトラックなどの高次の処理を処理系が備えているためにパターンマッチや探索を要するプログラムを簡単に書くことができます．このような利点を利用して，自然言語処理やエキスパートシステム，データベースなどで大規模なプログラムがつくられています．

要約すると，Prologによるプログラミングの利点は「読みやすく強力なプログラムを速く記述できる」ということになります．

エピローグ

以上ごく簡単にPrologを紹介しました．ここで紹介したのはほんの一端にすぎず，まだまだいろいろ強力な機能があります．最近はPrologの本もかなり出ているので興味を持たれた方はそれらを参照してください．Prologはほかの言語とはまったく異なるスタイルをもっているので，プログラミングをしているときにほかの言語では得られないような発想を思いつくことがあります．この辺がPrologプログラミングの醍醐味だと思います．

OCCAMとトランスピュータ

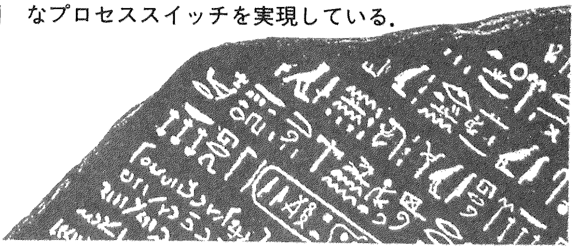
OCCAMの名称はPascalやAda同じように人名からきている．その人は「存在は必要以上に増やしてはいけない」という言葉を残した，14世紀の英国哲学者William Occamである．OCCAMは，オックスフォード大学計算学教授のCharles Antony Richard Hoare博士のCSP（Communication Sequential Processes）「Hoar 78」をベースに，英国INMOS社によって開発された（INMOS社はSRAMやDRAMなどを生産する国営エレクトロニクスメーカー，1984年に民営化された）．

OCCAMはトランスピュータを意識した並行プロセス記述と通信・同期機能および単純さを特徴とする言語である．単純さが特徴のひとつになっているのは，CSPの提案者であるC.A.R.Hoareの考えであると同時に，この言語で記述されたプログラムは複数のトランスピュータによって実行されることを意識して設計されているためである．いろいろな機能をトランスピュータ上で実現しハードウェアが重くなるのを避けたのである．また逆にこの単純さがOCCAMに多くの制約を生んでいる部分でもある．

並行プロセス間の同期方法はには変数共用による同期，メッセージ伝達による同期，通信チャンネルによる同期と大きく分けての３つの方法があり，それぞれにメリットやデメリットがある．OCCAMの並行プロセス間同期方法にはメッセージ伝達による同期方法が採用されている．

トランスピュータ（Transputer）とは，1982年にINMOS社が開発した通信機能を装備したマイクロコンピュータに，電子回路の要素としてトランジスタが多数利用されるように，コンピュータシステムの中核要素として数多く利用されることを願ってつけた名前である．

高速メモリと通信・同期機能を内蔵したマイクロコンピュータであるトランスピュータは，並行プロセスと通信の記述機能をもつ言語OCCAMを，いわばそのアセンブリ言語として見ることができる．見方を変えれば，OCCAMを効率よくインプリメントできるようにトランスピュータは設計されているわけである．トランスピュータはOCCAMの複数並行プロセス実行のためのスケジューラを具備し，これにより各並行プロセスがプロセッサ時間を共有できるようにしている．また，少数のレジスタと高速の内蔵RAMにより，高速なプロセススイッチを実現している．



プログラミング用語
メモリモデル　インテル社の80x86CPUなどのようにセグメントという概念から派生したメモリの管理方法

処理系紹介

Prolog

IF/Prolog Ver.3.4

UNIX, VMS, MS-DOSをはじめとする多くのOSと広範囲のハードウェアメーカーの製品で利用できるProlog処理系．どの機種についても均一の開発環境とランタイム環境を提供している．W.F.ClocksinとC.S.Mellishが「Programming in Prolog」で述べているプロシジャボックスモデルに基づくフルスクリーンデバッガが付属．ほかの言語からの呼出しや，その逆のインターフェースも装備．

製品名：IF/Prolog Ver.3.4 MS-DOS用
価格：150,000円
製品名：IF/Prolog Ver.3.4 MS-DOS80386CPU用
価格：300,000円
問い合わせ先：インタフェイスコンピュータジャパン(株)
03-3818-5826

Arity Prolog V5

米国Arity社製Prologインタプリタ＆コンパイラ．DEC-10 Prologに準拠．C言語で作成された関数をダイレクトにProlog内から呼び出すことができるなど，C言語と強力なインターフェースをもつ．NEC PC-9800シリーズ版にはグラフィック，サウンド，マウス，プリンタ制御を行う多目的ツールライブラリが付属．

NEC PC-9800シリーズ，富士通FMRシリーズ，松下Panacom Mシリーズ，東芝J-3100シリーズ，IBM PS/55シリーズなど各社パソコンに対応．ほかにOS/2対応版もある．

製品名：Arity Prolog V5 MS-DOS版
価格：98,000円
製品名：Arity Prolog PC-9800多目的TOOLキット付
価格：118,000円
製品名：Arity Prolog OS/2版
価格：100,000円
問い合わせ先：(株) ライフボート
03-3293-4711

Advanced RUN/PROLOG

DEC-10 Prologに準拠したPrologインタプリタ．Arity Prologとの互換性がある．プログラミング言語の学習環境を提供することを目的としたライフボート社のAdvanced RUNシリーズのひとつ．自然言語解析パーサDCGの搭載による文法解析機能，専用の対話型デバッガ，4MBの仮想データベース機能などをもつ．

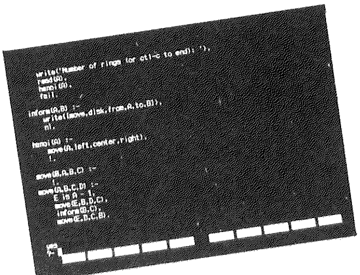
NEC PC-9801シリーズ，富士通FMR50/60/70，FM TOWNS，東芝J-3100シリーズ，IBM PS/55シリーズに対応．

製品名：Advanced RUN/PROLOG
価格：29,800円
問い合わせ先：(株) ライフボート
03-3293-4711

Prolog-KABA

パソコン用の日本語対応Prologインタプリタ．EMACS準拠のスクリーンエディタを内蔵．日本語マニュアル付き．NEC PC-9800シリーズをはじめ各社のパソコンに対応．

製品名：Prolog-KABA (NEC PC-9800シリーズ用)
価格：80,000円
問い合わせ先：岩崎技研工業 (株)
075-602-7878



Arity Prolog

Astral Prolog

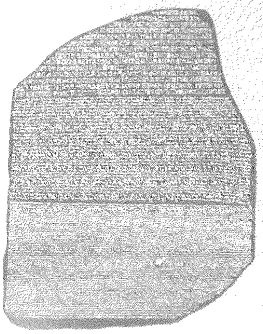
パソコンとワークステーション用に開発されたグラフィックと日本語対応のPrologインタプリタ．Prolog-Jの上位互換．ミニバージョンは浮動小数点演算，大容量メモリサポート等の機能が含まれない．

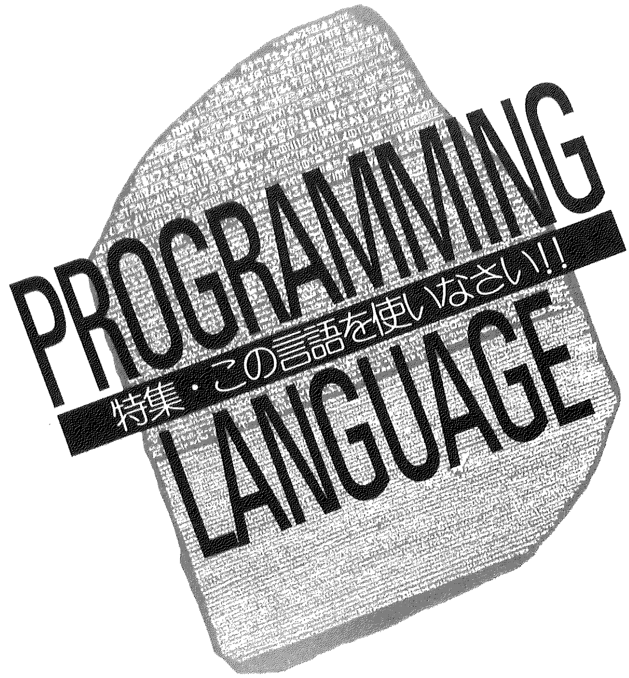
製品名：Astral Prolog (PC-9801用) フル・バージョン
価格：70,000円
製品名：Astral Prolog (PC-9801用) ミニ・バージョン
価格：6,000円，7,000円
問い合わせ先：(有) インターメディア・アクセス
03-3811-4084

Prolog-J

パソコン用に開発された日本語対応Prologインタプリタ．スクリーンエディタ内蔵．

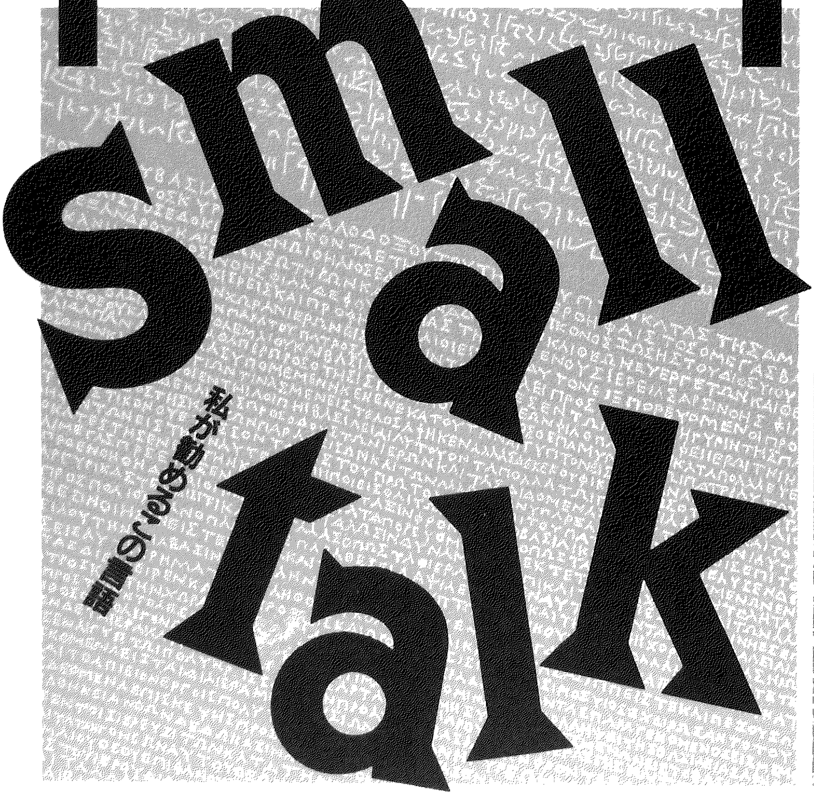
製品名：Prolog-J MS-DOS用
価格：33,000円
問い合わせ先：(有) インターメディア・アクセス
03-3811-4084





柔軟性のあるオブジェクト指向言語

Smalltalk　株式会社サザンパシフィック　近江武一



Altoの上でしか動かなかった

Smalltalkは，プログラミングに新しい概念とビジョンを持ち込みました．かなり毛色の変わったプログラミング言語ですが，はじめに目的があり，その目的を達成するために作られたという過程は，ほかのものと変わりません．ただ，その目的がほかの言語と少し違っていたのです．忘れてはならないのは，「Smalltalkはただのプログラミング言語ではない」ということです．プログラミング言語としての側面ももっていますが，それは本質ではありません．Turbo PascalをはじめとするIntegretaed Development Environmentの元祖でもあるのです．Smalltalkは，その目的として掲げた旗が非常に根元的なもので，なおかつ，その開発者達が妥協を知らぬ頑固ものだったために，なかなか通好みの優れものに仕上がっています．

Smalltalkの開発の発端は，「誰でも簡単に，効果的にコンピュータを使えるようにするにはどうしたらよいか」という，雲をつかむような研究です．1970年代のはじめのころ，Xerox社のPalo Altoリサーチ・センター(PARC)のLearning Researchグループが研究をはじめ，2年程度の間隔をおいてSmalltalk-72，-74，……，-80と開発を続けていきました．

それがSmalltalk-80として1981年にBYTE誌上で公開され，はじめて世にでました．大衆マイコン雑誌であるBYTE誌上でSmalltalkを取り上げるという試みも，なかなか大胆な話です．1981年というのはあのIBM PCがパソコン市場にさっそうと（？）参入した年です．CPUが8088，64KBメモ

プログラミング用語
予約語　識別子として使えないキーワード．ANSI Cでは32語が予約語になっている．

り，シングルフロッピー（5"）で3005ドルでした．この性能は，最悪ではとんど誰も相手にしませんでした．「同年に発売された話題作が，Osborne 1で1795ドルだった」といえば，当時のハードウェアのレベルがわかるでしょう．

なにしろ，当時の一般のパソコンでは，コンソールがテレタイプからキャラクタベースのディスプレイになり，フロッピーでCP/Mを使うのが多数派でした．その時代に，ビットマップディスプレイ，ハードディスク，Ethernet装備のシステムでしか動かないSmalltalk-80の話だけで1冊埋めつくしてしまったわけですから，大胆不敵な企画です．もちろんこのスペックの機械がこの時代に存在するわけはなく，最初のSmalltalk-80は，PARCで作られたAltoという実験機（今風にいうとワークステーション）で動いていました．そういう意味では，機械から作ったわけです．

オブジェクト指向を武器として

マイナーな割には，派手に登場したSmalltalkですが，そうなったのも，かなりそれまでのものと変わっていたからです．まず，見た目が変わっています．ソフトウェアにおいて「見た目」はなかなか重要で，巷で美人が得をするように，Lookに優れたソフトウェアは，それだけで得をすることも多いようです．さらに，「使いごこち」feelに優れていれば文句がないでしょう．

「見た目」で特徴的なところは，マルチウィンドウとマウスを使ったユーザーインターフェイスです．これは，「誰でも簡単に，効果的にコンピュータを使えるようにするにはどうしたらよいか」という目的のために，Smalltalkの開発者達が考え出した方法の1つで，たいへん独創的なものでした．私が思うに，この辺のことはプログラマにとっては，2次的に発生したSmalltalkの利点に過ぎないです．Smalltalkの本質は，オブジェクト指向という考え方を全面的にプログラムの世界に持ち込んできたことです．

私がSmalltalkを好きなのも，「オブジェクト指向」という自分の気に入った武器だけで巨大な敵に立ち向かう，いわばラ・マンチャの田舎者じみたところがあるからです．

Smalltalk/Vの出現

digitalkという米国の会社が，Smalltalk/Vというパソコン用のSmalltalkを販売しています．このSmalltalkはXelox社のSmalltalk-80とは違った経過をとっています．

digitalk社の創立者たち5人は，Smalltalkに惚れ込み，Olivetti社でZ8000をCPUとしたマシン上でSmalltalkを作りました．ちなみに，その機械は，Olivettiの経理用のオフコンとして日本にも入ってきています．

彼らはその後，独立してdigitalk社を設立，最初にIBM PC上で動くキャラクタベースのSmalltalkである「Methods」を作りました．1986年にはグラフィックをサポートしたSmalltalk/Vを，1988年にはプロテクトモードで動くSmalltalk/V 286，1989年にはSmalltalk/V Macとプレゼンテーションマネージャ用のSmalltalk/V PMを発売しました．今後も1991年にはMicrosoft Windows 3.0用のSmalltalk/V Windowsの発売を予定するなど，積極的に市場を展開しています．日本では，Southern Pacificが，1986年にEPSON PC-286，NEC PC-9801に「Methods」を，その後，Smalltalk/Vを1988年に移植しています．

Smalltalk/V 286を除く，そのほかの新しいSmalltalk/Vは（/V PM，/V Mac，/V Windows），移植の必要がほとんどないようです．日本で，新しく作業しなければならないところはNational Language Support Services関連のメソッドの追加と，エディタのDBCS対応だけだと思います．ただし，DBCSはデータとしてのみ扱うことを前提としています．ANSI Cでも，関数名にWide Charは認めていません．これから，OSやプログラミング言語のマルチバイト対応が流行ると思いますが，全体的な傾向として，予約語，キーワード，シンボルとしては，マルチバイト文字は許さないのが主流でしょう．

人間世界をモデルととらえる

ではなぜ，「オブジェクト指向」が選ばれたのでしょう．その背景には次のような考え方があります．

根本的に，コンピュータと人間はまったく異なったもので，日常の人間の考え方をそのままコンピュータのプログラムにすることはできません．裸に近いコンピュータを使うほど，コンピュータと人間の間のギャップは大きくなります．いまのコンピュータは本質的に人間と異質の存在だからです．

Smalltalkでは，人間が生活するこの世界を1つのモデルとしてとらえて，そのモデルをコンピュータの中に構築していきます．新しい考え方（モデル）でコンピュータを見ることによって，人間とコンピュータの間にある

プログラミング用語
ライブラリ　関数や手続きの集まりで，ソースファイルのヘッダで使用を宣言しコンパイル時，プログラムに取り込まれる．

本質的なギャップを埋めてしまおうというわけです．そして，Smalltalkは，この世界のモデルとして「オブジェクト指向」を採用しています．つまり，Smalltalkが「誰でも簡単に，効果的にコンピュータを使えるようにするにはどうしたらよいか」という目的地に到るために選んだ道が，「オブジェクト指向」の道だったのです．

コンピュータは，プログラミングができるのが最大の利点であり，プログラミングしなければ動かないというのが最大の欠点にもなっている厄介ものです．コンピュータにとってプログラミングは，両刃の剣というわけなのです．この問題に対する，1つの解答が，専用機としてのコンピュータです．たとえばワープロ専用機は，単独用途に限定してしまってプログラミングによる柔軟性を排除することによって，わかりやすさ，使いやすさを大幅に向上させています．ただし，このシステムは，コンピュータのもつ最大の利点を捨ててしまっています．

柔軟性のあるプログラミング

Smalltalkは，この問題を解決し，プログラミングすることによる柔軟性と，ほとんどプログラミングしなくていい簡便性を味あわせてくれます．

Smalltalkは，歴史的には上から下に降りてきたシステムですが，パソコンユーザーにとっては，複雑になって簡単にはプログラミングを楽しめなくなってしまったマイコンを，また自分の手元に引き寄せるものとなってくれます．ほとんどのプログラマは，グラフィカルユーザーインターフェイス(GUI)を内心苦々しく感じていると思います．「ちょっと見た目をよくするために，おぼえることばかり増えてプログラムは複雑になるし，デバックはしづらいし，何もいいことはない！」，こんなふうに思っている人も多いことでしょう．

Smalltalkは，その歴史的な成立ちからいってもわかると思いますが，この手のことは非常に得意です．Smalltalkは，その性質上短いプログラムのリストを載せることは，あまり意味がないのですが，たとえばいちばん短いウィンドウを開くプログラムは，図1

図1

```
'ようこそSmalltalk/Vの世界へ' edit
```

のようなものです．

これは，Cでいえば printf ("Hellow World")；プログラムのようなもので，実用性はまったくないものですが，Smalltalkではこれでウィンドウを開いてメッセージを表示します．これを少し発展させて，ウィンドウに題をつけて特定の場所に表示するとなると次のようになります．

図2

```
TextWindow
  windowLabeled: 'Smalltalk/Vの世界'
  frame: (0 @ 0 extent: 400 @ 100).
```

なかなか簡単なものでしょう．ここでSmalltalkの柔軟性を見せるためにSmalltalkのエディタの拡張に挑戦してみましょう．日本には，特有の「漢字」という問題がありますから，これを扱ってみましょう．Smalltalkのエディタ上で漢字変換を試みてみました．そのための第一歩は，入力した文字の範囲を指定し，メニューから変換を選ぶとその指定範囲が変換後の文字で入れ替わるという簡単なものです．

漢字変換には，2つの大きな関門があります．1つが文節の切りわけ（品詞解析）で，もう1つが高速な辞書アクセスです．ここでは，とても「文節の切りわけ」の問題までは手におえませんから，手抜きをして熟語変換でがまんします．そうすれば，問題は辞書の検索だけになるわけです．Smalltalkには便利なクラスがたくさんあるのですが，ここではその名もずばりDictionary classを使うと簡単に用がたせます．基本的に，このような考え方でプログラムを作ると，リスト1のようになります．

これは，非常に簡単で，「TextPane」のクラスメソッド「editMenu」に，Henkanというメニューを追加し，このメニューが，選択されたときに実行されるメソッド「henkan」を「TextPane」のインスタンスメソッドとして，新しく作っただけです．メソッド「henkan」は，KanjiDictionaryのなかを，選択された領域に入っている文字列をキーに検索し，その結果で，選択領域を置換しています．この方法は，Smalltalk丸抱えですべてやってしまおうという方針です．そのために，通常漢字フロントエンドプロセッサがもっている漢字辞書の部分や，品詞解析の部分をSmalltalkですべてもたなければなりません．ただし，最初は漢字辞書のアクセスや品詞解析の部分を今回のもののように簡単なものにしておいて，その後段々と本格的なものにすることができます．つまり，トップダウン的にプログラミングできるわけです．また，1度できあがったものを変更，改良することも簡単です．

このほかに，漢字フロントエンドプロセッサをSmalltalkの中から呼び出

プログラミング用語
リダイレクト　標準入力や標準出力先を変更すること．普通は標準入力はキーボード，標準出力はディスプレイが指定されている．

す方法が考えられます．この方法を使えば，Smalltalk側の負担は軽くなりますが，漢字辞書のアクセスや品詞解析の部分のアルゴリズムを変更，改良することはむずかしくなります．

これから，いろいろな拡張が考えられると思いますが，Smaklltalkならば，考えついてからそれを実現するのにそれほど時間はかかりません．その理由は，豊富なライブラリが揃っていることと，ライブラリモジュールの再利用性の高さです．この生産性の高さが，柔軟性と結びついたときにコンピュータは強力な道具となるのです．

(注) 今回のコードは，すべてSmalltalk/V Windows用です．これは，Smalltalk/V PMと非常に互換性が高いようで，プレゼンテーションマネージャとWindowsの共通のプラットホームとして注目できます．

リスト1

追加したメソッド

```
!TextPane methods !

henkan
        "Private - Kanji Henkan"

    self hideSelection.
    priorText := self selectedString.
    priorSelection := selection.
    newSelection :=
        self
            replaceWithText:
                (KanjiDictionary at:(self selectedString)).
    modified := true.
    self
        selectAfter: newSelection corner;
        forceSelectionOntoDisplay! !
```

追加

```
!TextPane class methods !
editMenu
        "Answer the Edit menu."
    | ctrlBits |
    ctrlBits :=  AfChar | AfControl.
    ^Menu new
        appendItem: '&Undo   Alt+BckSp'
            selector: #undo accelKey: nil  accelBits: AfAlt | AfVirtualkey;
        appendItem: 'Cu&t      Shift+Del'
            selector: #cutSelection accelKey: nil accelBits: AfShift | AfVirtual
key ;
        appendItem: '&Copy    Ctrl+Ins'
            selector: #copySelection accelKey: nil accelBits: ctrlBits | AfVirtu
alkey ;
        appendItem: '&Paste    Shift+Ins'
            selector: #pasteSelection accelKey: nil accelBits: AfShift | AfVirtu
alkey;
        appendItem: 'C&lear     Del' selector: #clearSelection ;
        appendItem: 'Select &All     Ctrl+A'
            selector: #selectAll accelKey: $a accelBits: ctrlBits;
        appendItem: 'Print &Selection     Ctrl+P'
            selector: #printSelection accelKey: $p accelBits: ctrlBits;
        appendSeparator ;
        appendItem: '&Find/Replace...     Ctrl+F'
            selector: #findReplace accelKey: $f accelBits: ctrlBits;
        appendItem: 'Find A&gain     Ctrl+G'
            selector: #again accelKey: $g accelBits: ctrlBits;
        appendItem: 'H&enkan     Ctrl+Z'
            selector: #henkan accelKey: $z accelBits: ctrlBits;
        title: '&Edit' ! !
```

これが追加される（appendItem: 'H&enkan ... の行）

PL/I

PL/Iは「Programming Language/I」の略で，IBM社が自信をもって世に送り出した言語である．科学技術計算から事務処理までを自由にこなし，アルゴリズムの記述にも適しているという，まさに「万能言語」である．PL/Iは1963年にIBMのユーザーグループの手によって開発が始まり，ALGOLの処理系を基にFORTRANやCOBOLの機能を取り込んで，1966年に完成した．

PL/Iは全ての言語の仕様を取り込んでいたのでALGOLやCOBOL，そしてFORTRANのプログラマは新たな知識なしにPL/Iを使うことができた．また，PL/Iはプログラムの生産性向上のためにいくつかの特徴をもっている．ひとつは言語仕様には予約語が存在しないことである．コンパイラ自身が適切に変数・関数・命令などを判断してくれるのである．また，あつかう変数の型指定の細かさも群を抜いている．ほかにも独特の機能に「割り込み」がある．ただし，これはアセンブラやC言語でいう「割り込み」とは別機能である．このPL/Iの影響を引き継いだのは唯一，BASICの割り込み機能である．

フルセットのPL/Iは，とてつもなく巨大なコンパイラとなった．大型コンピュータであれば問題はないが，ミニコンクラスのマシンでは，コンパイラを動かすだけで負担が大きすぎて，アプリケーションプログラムの大きさや，コンパイル速度などに制限が生じてしまうのである．サブセットのPL/Iも存在するが，それにはPL/Iのメリットは何も残ってはいない．

処理系紹介

Smalltalk

Smalltalk/V

Smalltalk/Vは，米国digitalk社が開発したパソコン上で高速に動作するSmalltalkシステムである．本来ワークステーション上の処理系(システム)であったSmalltalkをパソコン上に実現し，さらに低価格でユーザーに提供した実績は高く評価されている．日本国内においては，株式会社サザンパシフィックが日本語化および販売を行っている．

現在発売されている製品としては，Smalltalk/V，Smalltalk/V 286，Smalltalk/V Mac，Smalltalk/V PM，アプリケーションプログラム集として，Goodies #1〜#3，Smalltalk Communication Kitがある．

●Smalltalk/V

Smalltalk/Vは，NEC，PC-9801シリーズ，EPSON286シリーズ，またはIBM-PCのMS-DOS上で動作するシステムで，必要とされるハードウェアは，640KBのメインメモリとフロッピーディスクドライブ2台のみである．

Smalltalk/VはMS-DOS上の処理系であり，Smalltalk/Vの内部からDOSのコマンドを走らせたり，DOSのテキストファイルを読み書きしたりすることができる．また，アセンブラやCなどで書かれたプログラムを内部から呼ぶことができるので，ハードウェアを直接制御することも可能となっている．

●Smalltalk/V 286

Smalltalk/V 286は，IBM-PC ATまたはその互換機の80286/80386プロテクトモードで動作する，より高速・高機能なシステムである．

●Smalltalk/V Mac

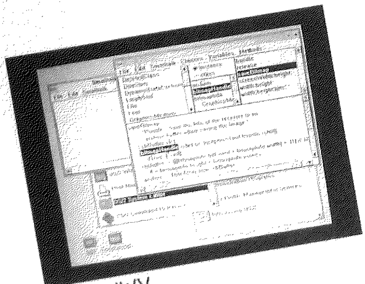
Macintosh上で動作するシステムで，ユーザーインターフェースはMacintoshの標準インターフェースに準拠している．

●Smalltalk/V PM

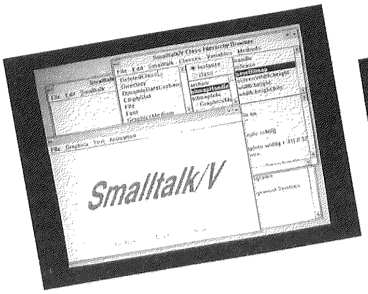
Microsoft OS/2 Presentation Manager version 1.1以降で動作するシステムで，史上初のSmalltalkコンパイラである．OS/2 PM環境に完全にコンパチで，PMの内部に立ち入ることなくそのすべての機能を利用できる．このためプログラマは，PMの細部に惑わされることなく自身のアプリケーション開発に集中することが可能となる．

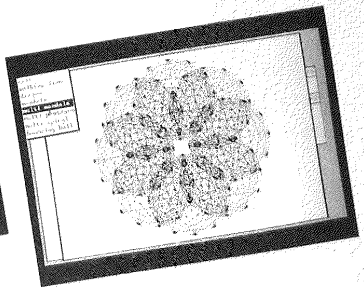
Microsoft社のビル・ゲーツもSmalltalk/V PMを，優れたOS/2 PM用のアプリケーション開発システムとして推奨している．

製品名：Smalltalk/V
価格：49,800円（PC-9801用）
　　　39,800円（IBM PC AT用）
製品名：Smalltalk/V 286
価格：69,800円（IBM PC AT用）
製品名：Smalltalk/V Mac
価格：49,800円（Macintosh用）
製品名：Smalltalk/V PM
価格：98,000円（OS/2 PM用）
製品名：Goodies #1〜#3
価格：各16,800円
製品名：Smalltalk Communication Kit
価格：16,800円
問い合わせ先：㈱サザンパシフィック
　　　045-314-9514



Smalltalk/V





プログラミング用語
ローカル変数　局所変数とも呼び，関数内でのみ有効となる変数

Smalltalk-80 System Package

Smalltalkが，ゼロックス社のパロアルト研究所でアラン・ケイらによって開発されたことはよく知られている．Smalltalk-80 System Packageは，このSmalltalkの本家ともいうべきゼロックス社から発売されているパソコンおよびワークステーション用の開発システムである．

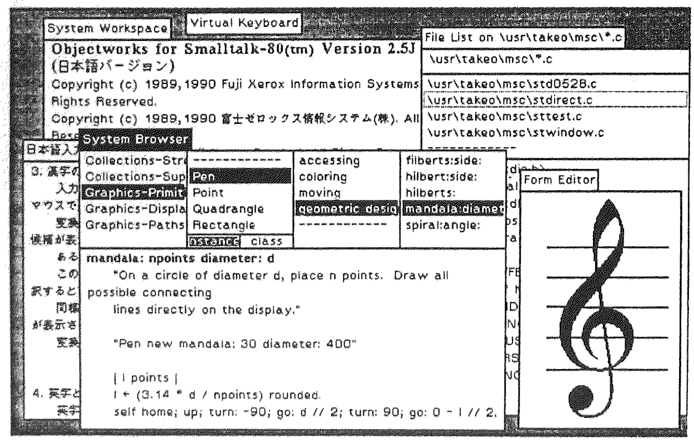
Smalltalk-80 System Packageは，仮想イメージ（Virtual Image）と仮想マシン（Virtual Machine）から構成されている．仮想マシンは，ハードウェアやOSと仮想イメージを結び付けるためのもので，仮想イメージにあるコードを高速に解釈・実行する．仮想イメージは，Smalltalk-80自身で記述されたSmalltalk-80自身のシステムで，開発ツールやウィンドウシステム，さらにユーザーアプリケーションシステムを含んでいる．

Smalltalk-80 System Packageでは，仮想イメージごと同じバージョンのSmalltalk-80を使用している他のプラットホームに移植することができるため，異機種間でバイナリレベルの互換性を保持することが可能となる．

また，ランタイム開発環境を備えているため，デバッガなどの開発ツールをアプリケーションから切り離してデリバリーシステムを作成することもできる．

80386CPUをもつNEC PC-9801シリーズおよびIBM PCとその互換機，Macintoshで動作する．それぞれ日本語対応版と英語のみの製品がある．

製品名：Smalltalk-80 System Package
価格：140,000円（PC-9801，Ver.2.5J日本語機能つき）
120,000円（PC-9801，Ver.2.5）
140,000円（386MS-DOS，Ver.2.5J日本語機能つき）
120,000円（386MS-DOS，Ver.2.5）
140,000円（Macintosh，Ver.2.5J日本語機能つき）
120,000円（Macintosh，Ver.2.5）
問い合わせ先：富士ゼロックス情報システム㈱　03-3378-8011



Smalltalk-80 System Package

Smalltalk-80/VM68版

Smalltalk-80/VM68版は，PC-9801（E，F，M，VF，VM，VX対応）の拡張スロットにMC68000搭載のCPUボードおよび1MBのメモリボード2枚を挿入することで動作するフルセットのSmalltalk-80開発システムである．

CPUボードを使用していることからもわかるように，Smalltalk-80/VM68版はPC-9801のディスプレイやキーボードなどを借りたミニコンシステムというべきしろものである．実際，Smalltalk-80/VM68版で開発されたプログラムは，ゼロックス社の専用機のものと互換性がある．

また，Smalltalk-80/VM68版の特徴として，世界で唯一リアルタイムで画面のハードコピーが取れることがあげられる．

製品名：Smalltalk-80/VM68版
価格：620,000円（MC68000CPUボード×1，1MBメモリボード×2を含む）
問い合わせ先：㈱エス・アンド・ティー　03-3754-2367

BOOK LIST

プログラミング言語をもっと詳しく知るために

　プログラミング言語を習得するには，それなりの時間と労力を必要とする．新たな世界への第一歩が言語処理ソフトウェアのパッケージから出したてほやほやのマニュアルから始まるのもいいが，よい本はときとしてすばらしい道しるべとなってくれるものだ．

　ここでは，プログラミング言語全般と個々のプログラミング言語について，より詳しい情報と知識を得るための書籍を紹介しよう．ここにあげた本は教科書的なものが中心だが，なかには設計者や開発者によって書かれたその言語の「バイブル」もある．プログラマたるもの書名だけでも心にとめておいても悪くはないのではないだろうか．

Assembler/BASIC

　アセンブラ，BASICはどちらも実際にその言語を使用するハードウェアと深い関わりのある言語である．対象とするCPU，実際に利用する処理系に即して書かれた本を熟読するのが望ましい．

C

「プログラミング言語C－UNIX流プログラム書法と作法」
B.W.カーニハン & D.M.リッチー著，石田晴久訳／共立出版／2,575円

「プログラミング言語C　第2版－ANSI規格準拠」
B.W.カーニハン & D.M.リッチー著，石田晴久訳／共立出版／2,800円

　Cについては今さらいうまでもない定番．UNIXやCの開発者自身によるCの最初の参考書で著者の頭文字をとってK&Rと呼ばれている．現在は1988年末に提出されたC言語のANSI規格に基づいて書き直された第2版が出ている．

「C　アンサー・ブック　第2版」
C.L.トンド・S.E.ギンベル著，矢吹道郎訳／啓学出版／2,000円

　K&R中の演習問題の解答を示した本．K&Rを最初から勉強していることを前提にした解答集なので，まだ勉強していないテクニックを使った解答を載せるようなルール違反もない．

C＋＋

「プログラミング言語C＋＋」
B.ストラウストラップ著，斉藤信男訳／トッパン／4,071円

　C＋＋の設計者であるAT&Tベル研究所のストラウストラップが自ら書き下したC＋＋プログラミングの定本．

「C＋＋言語入門」
S.C.デューハスト & K.T.スターク著，小山裕司訳／アスキー／2,300円

　基本的なプログラミング方法からオブジェクトの設計方法まで段階をふんで解説したC＋＋言語の入門書．ATTの2.0版に準拠している．

COBOL/FORTRAN

　商用として早くから使われてきたこれらの言語は，パソコンでこそ一般的ではないが，もっとも多くのコンピュータ上で利用できるといってもいいだろう．プログラムを作成するときに最も頼りになるのは，結局のところその機種のユーザーマニュアルと文法解説書なのだが，COBOL，FORTRANについてはその傾向が最も顕著である．

「電子計算機プログラミング言語COBOL」（JIS X 3002 1988）
日本規格協会／12,000円★

「電子計算機プログラミング言語FORTRAN」（JIS C 6201 1982）
日本規格協会／4,500円★

　COBOL，FORTRANにはいわゆるバイブルは存在しないが，現在日本でこれらの言語の標準はJISによって定められている．言語処理系の作成に携わるのなら一度は目を通しておきたい．

Hypertalk

「ザ・ハイパーカード」（上）（下）
D.グッドマン著，プロジェクトハウス訳／ビー・エヌ・エヌ／各3,300円

　ハイパーカードの詳細な解説書であるTHE COMPLETE HYPERCARD HANDBOOKの邦訳．

LOGO

「マインドストーム」
S.パパート著，奥村喜世子訳／未来社／2884円

　LOGOの設計者のひとりパパートの著書．

「LOGO－エーベルソンのLogo入門」
H.エーベルソン著，五藤博義訳／啓学出版／2,680円

　LOGOの開発者のひとりであるエーベルソンによって書かれた基本的なLOGOの入門書．タートルグラフィックスの解説，リスト処理の方法，再帰の使い方などLOGOを使う上で必要な内容を詳細に説明している．

Lisp

「LISP」（情報処理シリーズ4）
P.H.ウィンストン & B.K.P.ホーン著，白井良明・阿部憲応訳／培風館／3,708円

　Winstonの「Artifical Intelligence」の第2部を改訂した「LISP」の翻訳．Lispプログラミングの紹介とLispの実際の使用例の紹介．

Prolog

「Prologプログラミング」
W.F.クロックシン & C.S.メリッシュ著，中村克彦訳／マイクロソフトウェア／6,180円

　原題は「Programming in Prolog」．Prologを実際的なプログラミング言語として教育するための指導書として著された．

Pascal/Modula-2

「PASCAL」（情報処理シリーズ2）
K.イェンゼン & N.ヴィルト著，原田賢一訳／培風館／2,369円

「アルゴリズム＋データ構造＝プログラム」（コンピュータサイエンス研究書シリーズ40）

N.ヴィルト著，片山卓也訳／日本コンピュータ協会／6,180円
「Modula-2プログラミング」(改訂第3版)
N.ヴィルト著，斎藤信男訳／マイクロソフトウェア／6,180円
Pascal，Modula-2両言語の設計者であるN.ヴィルトは，それぞれの言語についてバイブルともいうべき本を著している．「PASCAL」は言語の学習者，利用者，処理系制作者を，「Modula-2プログラミング」は言語の学習者を対象とした本．「アルゴリズム……」はプログラムを本格的に勉強しようという人向け．

Smalltalk

「Smalltalk-80言語詳解」
A.ゴールドバーグ＆D.ロブソン著，相磯秀夫監訳／オーム社／7,004円

「Smalltalk-80—対話型プログラミング環境」
A.ゴールドバーグ原著，相磯秀夫監訳／オーム社／5,665円

「基礎からのSmalltalk-80—オブジェクト指向のプログラミング」
飯野昌一・及川一成・田制貴俊・川西真木著／工学社／3,000円★

「オブジェクト指向とSmalltalk」
小林史典著／CQ出版／1,700円

プログラミング言語全般

何かひとつの言語でプログラムを作成した経験があって，当面ほかの言語ではさしせまったプログラミングの必要がないのなら，プログラミング言語全般について解説した本を勧めたい．

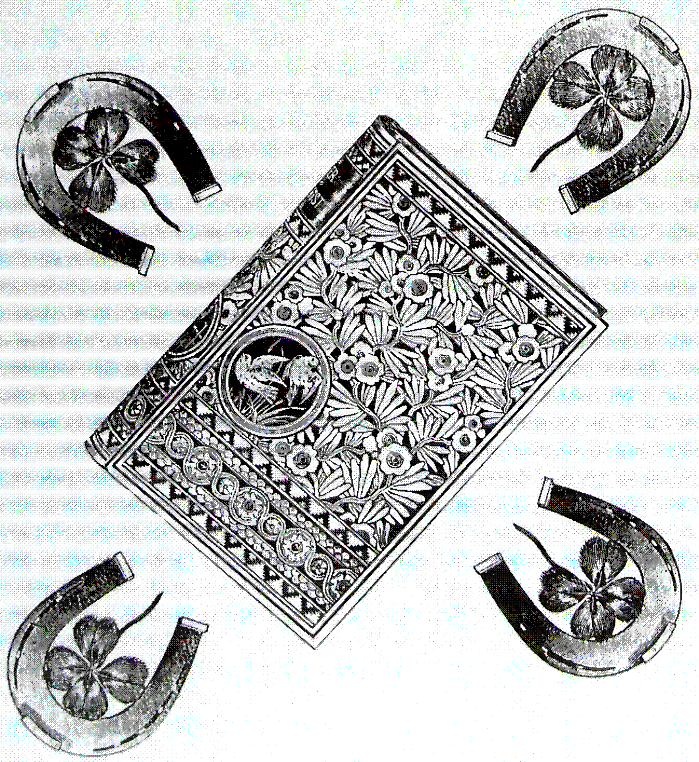
「プログラム言語」(岩波講座情報科学9)
米田信夫編／岩波書店／2,500円★
プログラム言語の意義と役割，記述法，個々のプログラム言語の特徴，コンパイラの仕組み，ソフトウェアのライフサイクルなどの多彩な内容を盛り込んだ「プログラム言語論」ともいうべきアカデミックな1冊．COBOL，FORTRAN，PL/I，Adaについてはそれぞれ独立した章を設けて解説している．Lisp，C，APL，RPGなどについても言及している．
同シリーズにはこのほかにも「3 プログラムの読み方」があり，BASICやPascalについてはこちらで詳しく解説している．

「プログラミング言語の新潮流」(計算機科学／ソフトウェア技術講座2)
井田哲雄著／共立出版／2,884円
COBOLやFORTRAN，Pascalなどの手続き型言語に対して，関数型および論理型言語といわれるLisp，Scheme，KRC，FP，Prologを取り扱っている．

「パソコン言語学」
石田晴久監修／アスキー出版局／1,800円★
パソコンで利用できる処理系がある9つの言語(COBOL，FORTRAN，Pascal，C，Ada，LOGO，Prolog，Lisp，FORTH)について，それぞれの言語に精通した9人の著者がその成立過程，言語の概要，文法，プログラム例を紹介する．まったく同じ機能を持った金種計算プログラムをそれぞれの言語で記述した例を紹介しているので，それぞれの言語の書き方を具体的に比較することができる．

「コンピュータ言語進化論—思考増幅装置を求める知的冒険の旅」
H.レヴァイン＆H.ラインゴールド著，椋田直子訳／アスキー／1,900円★
アリストテレスの三段論法からプログラミング言語の未来にいたるまで論理的な思考を形式化する方法の発展をたどって，思考を増幅するための道具いう観点からプログラミング言語をとらえなおした一冊．個々のプログラミング言語の文法を知るというよりも，読物としておもしろい．

(★の書籍は価格が消費税別)



祐安重夫

プログラマの道具学

第3回 モデム

Illustration : Mikio Sasaki

パーソナル・コンピュータが出現するはるか以前から，きたるべき情報化社会を予測するといったこころみのなかでは，通信回線によって大型コンピュータとむすばれた端末機が家庭にはいりこむことで，家で自由にコンピュータ資源を利用できるようになるという未来像がえがかれてきた．その当時でも通信回線を通じてのコンピュータの利用はおこなわれていたが，個人が使用するにはあまりにも高価なものだった．

その頃は，端末機自身が電動タイプライタに通信機能を付加した程度の，きわめて単純なものだったが，現在のパーソナル・コンピュータよりはるかに高価だったことは，いうまでもない．もちろん技術がすすみそれが普及していくことで，コストはどんどんさがっていくし，また端末機の機能も高度なものになっていくことは予想できた．

だがマイクロプロセッサが発明され，パーソナル・コンピュータがつくられることで，端末機ではない独立したコンピュータが家庭の中にはいりこむようになった．それまでは端末機が高度化してほとんどコンピュータ化したとしても，それはあくまでも端末機であり，中央のコンピュータとつながれることではじめて本格的に機能するというイメージがあったのだが，パーソナル・コンピュータはそれだけで独立したコンピュータであったのだ．

こうして，中央集権的な情報化社会のイメージが，個人のレベルではパーソナル・コンピュータによる草の根型のネットワークに転換していく契機がうまれたといえるかもしれない．

コンピュータを通信回線に接続するのは，技術的には簡単なことだが，日本では法律上の規制がかなりあった．中央集権的なネットワークには対応できたが，個人レベルでの回

線の利用にはほとんど応じられないという状況が，相当ながくつづいていた．

この段階で私自身は音響カプラを購入して，パーソナル・コンピュータを電話回線に接続した．もちろん現在のようにパソコン通信が存在するわけではなく，つなぐ先はJOISとかDIALOGといったデータベース・サービスだった．これらのサービスはパーソナル・コンピュータが出現する以前から存在していたものだが，パソコンによって専用の端末機を用意しなくても利用できるようになったことで，利用者は増加したのではないだろうか．

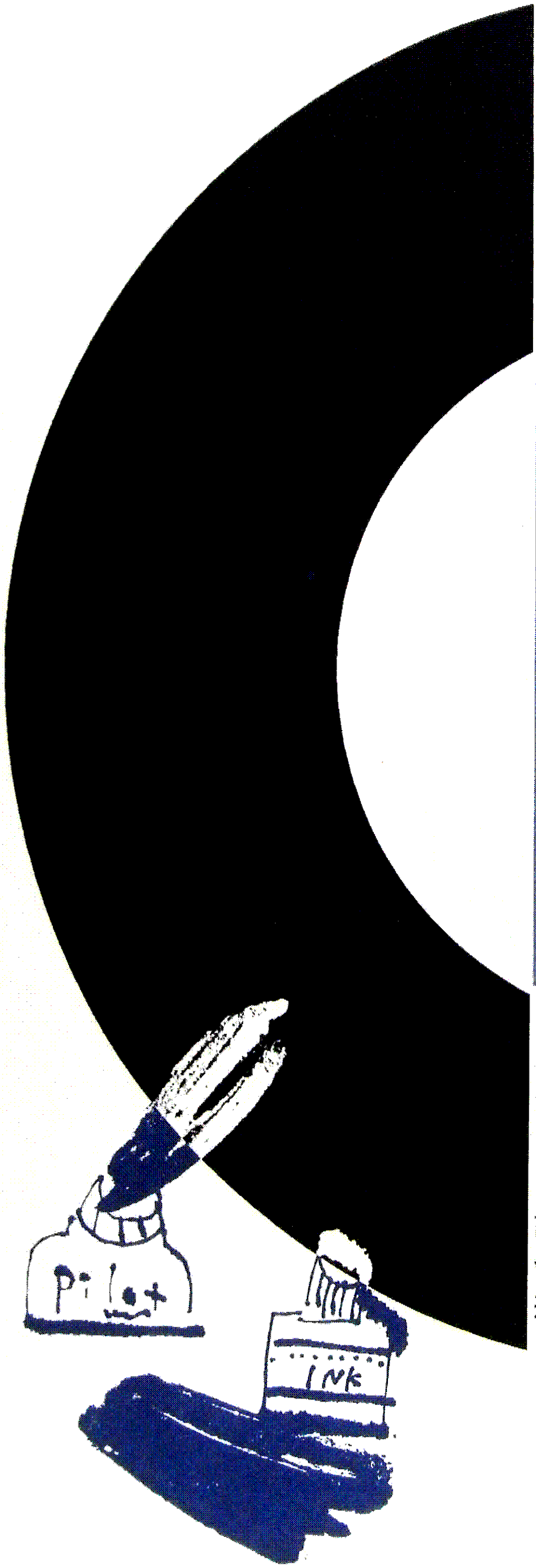
それからしばらくして，電話事業が民営化され，さらに1社独占ではなくなる方向になるとともに，回線の利用に関する様々な規制が解除されるようになってきた．これまでの300bpsの低速の音響カプラにかわって，1200bpsや2400bpsの比較的安価なモデムが市場にでてくるようになり，おおくのパーソナル・コンピュータが電話回線と接続されるようになった．

モデムのおもしろい点は，パーソナル・コンピュータを端末機として，ホスト・コンピュータに接続してサービスをうけるだけではなく，みずからホスト・コンピュータになれる機能を提供することである．それまでホスト・コンピュータといえば，大型コンピュータというイメージがつよかったのだが，家庭のパソコンが簡単にホスト・コンピュータになれるようになったのだ(もちろん，そのためのソフトウェアを用意する必要はあるが)．

こうして，草の根BBSとよばれるパソコン通信のホストが，あっというまに全国的にひろがっていった．また大手の商用パソコン通信サービスも，登場するようになったのである．現在モデムは，より高速化され機能もおおくなる傾向にある．なかにはホスト機能を内蔵したモデムつき電話などというものまで，登場するようになってきている．高速な物としては，HSTやTrailBlazerなどがあるが，これらは独自の方式を使用している．私がつかっているのはTrailBlazerの高速モデムで，最高18000bpsまでに対応するようになっている．

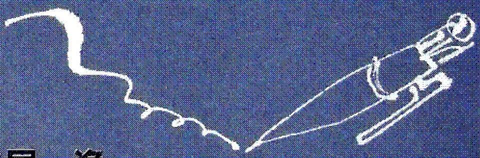
もっとも現在のところ一般に使用されているのは2400bpsまでの物がおおく，独自の方式を採用した高速モデムでも，2400bpsまでは相互に接続できる．また最近では9600bpsについても，標準化がおこなわれるようになってきており，高速モデムでも最新の製品は，自社の方式と標準方式の両方に対応するようになってきている．

Illustration : Masami Nozue



実践C塾 第3回

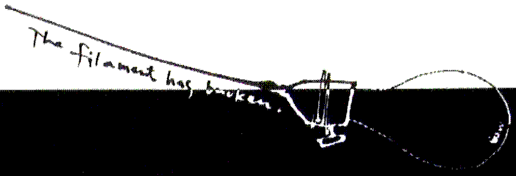
MS-CとTurbo Cのコード生成の特徴



塾頭 水居 浄

この記事では，MS-DOS上の代表的なC言語処理系であるMicrosoft CとTurbo Cについて，それらの特徴を細かく検討することによって，MS-DOS上で実際にプログラミングする際のさまざまな問題点，およびそれらの解決方法などについて論じていきたいと考えています．
前回は，メモリモデルとオーバーレイについてお話しました．今回は両コンパイラのその他のコード生成上の特徴について述べたいと思います．

ワンラインNEWS
TKCは富士通が昨秋に発売した無停電型コンピュータ「SURE SYSTEM 2000」30台の導入を決定した．同社は，このコンピュータのユーザー第1号となる．



1 ポインタ（pointer）

ポインタは，Cではもっとも重要な概念であるといえます．ポインタの保持する値は，メモリ空間上のアドレスそのものです．しかし，これまで述べてきたように，8086系CPUではnearアドレスとfarアドレスという概念が存在します．このため，ポインタにもnearポインタとfarポインタがサポートされており，さらにこれらの派生形としてのhugeポインタやbasedポインタなどがあります．

nearポインタ
オフセット部のみの演算で処理する

nearポインタは16ビットのポインタです．すなわち，16ビットのオフセットアドレスを保持します．セグメントはどこになるかというと，通常はDGROUPです．つまり，デフォルトのデータセグメントDGROUP内の領域を指すことしかできません．しかし，オフセット部のみの演算で処理が行われるため，一般に処理が高速です．ラージモデルなどの複数のデータセグメントをもつメモリモデルでも，nearポインタはDGROUP内を指します．

farポインタ
全メモリ領域を指すことができる

farポインタは，DGROUP以外の全メモリ領域を指すことができるようにしたポインタです．MS-DOSのアドレス空間は20ビットですが，farポインタは32ビット値で表します．これは，8086系CPUのアドレスが16ビットのセグメント値と16ビットのオフセット値の組み合わせで表されるため，20ビットの物理アドレス値に変換せずにそのまま扱ったほうが効率が良いからです．

farポインタは，上位16ビットにセグメント値，下位16ビットにオフセット値を保持しますので，**図1**のようにlong型の定数を使って絶対アドレスを代入（もしくは初期化）することができます．

この場合，セグメント値が0xA000，オフセット値が0x0000のfarポインタになるわけです．キャスト演算子を付けないと，コンパイル時に警告またはエラーを生じます．

実際には，farポインタの値の取得と設定に使用できるマクロが，Microsoft C，Turbo C双方に用意されています．

Microsoft Cでは，**図2**のようなマクロがdos.hで定義されています（Ver. 5.1）．

これは，farポインタをセグメント部とオフセット部に分解するマクロです．さっきのfarポインタvramをこれに与えると，FP_SEG(vram)の値は0xA000に，FP_OFF(vram)の値は0x0000になるわけです．このマクロの定義を注意深く見てみると，**図3**のように代入文の左辺に置いて，farポインタに値を設定する用途にも使用できることがわかります．

余談ですが，このVer. 5.1のFP_OFF，FP_SEGの定義には実は誤りがあります．普通のfarポインタの場合は問題ないのですが，たとえば**図4**のようなfarアドレスに置かれたfarポインタを与えると正常に動作しないのです．

このバグは，Ver. 6.0では**図5**のよ

図1　long型の定数で絶対アドレスを代入する例

```
short far *vram = (short far *)0xA0000000L;
```

図2　MS-C 5.1のdos.hのマクロ定義

```
#define FP_SEG(fp)  (*((unsigned *)&(fp) + 1))
#define FP_OFF(fp)  (*((unsigned *)&(fp)))
```

図3　代入文を左辺においてfarポインタに値を設定する例

```
FP_SEG(vram) = 0xA000;
FP_OFF(vram) = 0x0000;
```

図4　MS-C 5.1では正常に動作しなかった

```
char far * far fp;
```

図5　MS-C 6.0では次のように修正された

```
#define FP_SEG(fp)  (*((unsigned _far *)&(fp)+1))
#define FP_OFF(fp)  (*((unsigned _far *)&(fp)))
```

図6　Turbo C 2.0のdos.hの定義

```
#define FP_OFF(fp) ((unsigned)(fp))
#define FP_SEG(fp) ((unsigned)((unsigned long)(fp) >> 16))
```

図7　Turbo Cのfarポインタの値の設定

```
#define MK_FP(seg,ofs) ((void far *) ¥
    (((unsigned long)(seg) << 16) | (unsigned)(ofs)))
```

図8　farポインタに代入可能な値が作成できる

```
short far *vram = MK_FP(0xA000, 0x0000);
```

図9　Turbo C++でのマクロ定義

```
#define MK_FP(seg,ofs) ((void _seg *)(seg) + (void near *)(ofs))
#define FP_SEG(fp) ((unsigned)(void _seg *)(void far *)(fp))
#define FP_OFF(fp) ((unsigned)(fp))
```

図10　処理系に依存しないマクロ

```
typedef unsigned short WORD;
typedef unsigned long  DWORD;
#define HIWORD(x) (WORD)((DWORD)(x)>>16&0xFFFFL)
#define LOWORD(x) (WORD)((DWORD)(x)&0xFFFFL)
#define MAKEDWORD(hi,lo) ¥
  ((DWORD)(hi)<<16&0xFFFF0000L|(DWORD)(lo)&0xFFFFL)

#define FAR_SEG(p) HIWORD((void far *)(p))
#define FAR_OFF(p) LOWORD((void far *)(p))
#define MAKEFP(seg,off) (void far *)MAKEDWORD(seg,off)
```

うに修正されています．

_far というのは，Ver. 5.1までのfarと同等のキーワードです．Ver. 6.0からは先頭にアンダースコアを付けたfarという表記が標準になりました．これは，ベンダ独自の拡張した名前には先頭にアンダースコアを付けるべきであるという，ANSI規格のガイドラインに沿った変更です．

Turbo Cにも同じ名前のマクロがdos.hで提供されています．しかし，その定義は図6のようなもの(Ver. 2.0)で，Microsoft Cとは異なります．

見ての通り，このマクロではfarポインタのオフセット値，セグメント値を取り出すだけで，farポインタの値を設定することはできません．そこで，Turbo Cではfarポインタへの値の設定は図7のようなマクロでサポートしています．

このMK_FPに図8のようにオフセット値とセグメント値を与えると，farポインタに代入可能な値が作れるわけです．

なお，Turbo C++では，これらのマクロ定義は図9のように変更されています．

_seg というのはやはりTurbo C++で拡張されたキーワードですが，詳細については後述します．

さて，上記のようにMicrosoft CとTurbo CではFP_SEG，FP_OFFの定義が異なるわけですが，これは次の2つの点で移植性の問題を生じます．

① Microsoft Cの定義では，FP_OFF，FP_SEGにポインタ変数以外のものを与えることができない．すなわち，配列のアドレスなどを与えることができない．

②Turbo Cの定義では，FP_OFF，FP_SEGを使った代入の記述ができない．このため，Microsoft Cで書かれたプログラムにそのような記述があった場合は，Turbo Cでコンパイルするためには修正が必要になる．

どちらかの処理系に強く依存したプログラムを書く場合には，あまり気にする必要はありませんが，少しでも移植性を良くしたいという場合は，この問題を回避する方法はひとつしかありません．つまり，これらのマクロは使わずに，相当するものを自分で定義して使うことです．私の場合，図10のようなマクロを定義して使っています．

段階的に定義しているため少々ごちゃごちゃしていますが，要はTurbo Cのマクロとほぼ同じもので，キャストをより細かく行っているだけです．混乱を避けるため，標準のFP_OFF，FP_SEG，MK_FPとはあえて違う名前にしています．

farポインタの演算

演算はオフセット部のみで行われるため制約が生じる

前述のように，farポインタは上位16ビットにセグメント値，下位16ビットにオフセット値を保持する32ビットポインタです．しかし，ポインタの演算はオフセット部のみで行います．具体的には，ポインタと整数の加減算やポインタ同士の引き算といった演算は，オフセット部のみで計算されるのです．

こういう性質のため，farポインタには次のような制約が生じます．

①セグメント値の異なるfarポインタどうしの比較や演算は本質的に無意味である．

たとえば，図11の2つのfarポインタ

ワンラインNEWS
ソフトハウスのパーソナルメディアは同社のDBソフト「Let'sアイリス3.0」を企業向けにサイトライセンスでの販売を開始した．

は，MS-DOS上では同じ物理アドレスを指していますが，このif文の式は偽になります．

図11　このif文の式は偽になる

```
char far *a = (char far *)0x12340005L;
char far *b = (char far *)0x10002345L;

if (a == b) {
  .
  .
  .
```

これは比較の際に，同じ物理アドレスかどうかまでは調べないで，単純に値を比較するコードが生成されるためです．farポインタどうしの引き算の場合も同様ですから，a－bといった演算は無意味です．

2つのfarポインタが同じ物理アドレスを指しているかどうかを調べるには，**図12**のようなマクロで正規化してしまうのが良い方法です．ここでいう正規化とは，オフセット部を可能な限り小さな値(すなわち16未満)になるようにすることで,同じ物理アドレスを指すポインタは必ず同じ値になります．

図12　マクロで正規化を行った例

```
#define NORMALIZEFP(fp) ¥
  MAKEFP(FAR_SEG(fp)+(FAR_OFF(fp)>>4),FAR_OFF(fp)&0x000F)
```

こうすると，**図13**のif文は真になります．

図13　このif文の式は真になる

```
if (NORMALIZEFP(a) == NORMALIZEFP(b)) {
  .
  .
  .
```

②オフセット部が0xFFFFを超えた場合，ラップアラウンドしてしまう．

farポインタではオフセット部でしか演算を行いません．たとえば，**図14**のようなポインタで&a［0x100］はどこを指すかというと，0x1000:0x0000を指すことになります．

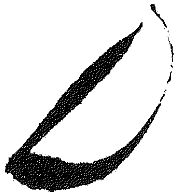
図14　オフセット部が0xFFFFを超えた場合

```
char far *a = (char far *)0x1000FF00L;
```

実際には，こういう風に物理アドレスの値を直接的にfarポインタに与えることは，VRAMなどの特殊な領域を除いては，普通やりません．farポインタが指すのは，通常は他の変数や配列のアドレスか，malloc等で動的に割り当てたヒープ領域のアドレスです．配列やヒープは，割り当てる時点で大きさが決ってますから，上記のようなラップアラウンドが起こらないようなアドレスが自動的に割り当てられます．つまり，普通の使い方をしている限りは，このラップアラウンドの問題はとくに気にする必要はないといえます．

hugeポインタ
セグメント部に対しても演算を行う

hugeポインタは，基本的にはfarポインタと同じ32ビットポインタで，上位16ビットにセグメント値，下位16ビットにオフセット値を保持します．しかし，ポインタの演算はオフセット部のみでなくセグメント部に対しても行います．すなわち，ポインタと整数の加減算やポインタ同士の引き算はセグメント部も含めて行われます．



MS-Cにおけるhugeポインタの制限

hugeポインタのインプリメントは，Microsoft CとTurbo Cでかなり異なります．Microsoft Cのhugeポインタは，データがセグメントバウンダリを超えたときにセグメント値を更新するという方式です．つまり，**図15**のようなとき，&hp［0x0F］は0x1000:0x2354，&hp［0x10000］は0x2000:0x2345となるわけです．このようなインプリメントになっているため，Microsoft Cのhugeポインタでは，次のような制限が生じます．

①正規化されていないため，farポインタの場合と同様に，2つのhugeポインタを直接比較・演算することはできない．

②128KBを超えるような配列領域を指すhugeポインタでは，配列の各要素は2のべき乗の大きさでなくてはならない．

これは，128KBを超える配列ではその中にセグメントバウンダリを2個以上もつことになってしまうため，各要素が64KBをちょうどの大きさで分割できるようにする必要があるためです．128KB未満の配列では，セグメン

図15　hugeポインタの例

```
char huge *hp = (char huge *)0x10002345L;
```

トバウンダリを1個に抑えることができるため，要素の大きさが何であっても先頭アドレスを適当に設定することでうまくバウンダリを要素が跨がないように調整することができます．

このようなバウンダリに関する制限があるため，hugeポインタが指す領域は，huge配列として宣言された領域か，もしくはhallocで取得した領域でなくてはなりません．これら以外の，たとえば_fmalloc等で取得した領域を与えると，予測できない結果に陥る可能性があります．

Turbo Cは常に正規化して演算する

一方，Turbo Cのhugeポインタは，演算するときには常に正規化されます．上記のhugeポインタhpの値は，初期化の直後では0x1000:0x2345ですが，比較など演算を行うときには正規化されて0x1234:0x0005として扱われるのです．したがって，&hp[0x0F]は0x1235:0x0004，&hp[0x10000]は0x2234:0x0005となるわけです．

この方式だと，Microsoft C方式のような制限は生じません．比較などの演算も簡単に正しく行うことができますし，要素の大きさに関する制限もありません．したがって，Turbo C方式のほうが優れているといえますが，その一方で性能という観点からは少々不利だともいえます．つまり，Turbo C方式では，hugeポインタは常に正規化されなくてはならないため，比較的小さなインクリメント処理を行った場合でも，正規化してセグメント値を更新しなくてはならなくなるケースが生じるからです．Microsoft C方式では，セグメントバウンダリを超えない限りセグメント値の更新が生じません．

とはいえ，Microsoft Cの②のような制限は少々不自由に感じられるのも事実です．たとえば，図16のようなhuge配列宣言は，128KBを超えているのに各要素が6バイトで2のべき乗ではないため，エラーになります．

図16　huge配列宣言の例

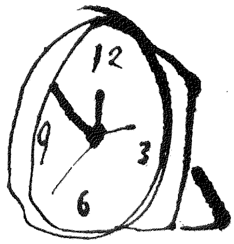
```
struct {
     short a;
     long  b;
} huge hs[30000];
```

プロテクトモード対応の問題

このような制限があるにもかかわらず，Microsoft Cがなぜこの方式を採用したのかというと，実はプロテクトモード対応という問題があるためだと思われます．MS OS/2やWindows 3.0などの新しいOS環境は，80286以降のCPUのプロテクトモードで動作します．プロテクトモードでは，セグメントレジスタの意味が少々変わります．80286アーキテクチャを説明するのは本稿の目的ではありませんから，詳細は省略しますが，要はもう1段階間接的になったと思えば良いのです．プロテクトモードでは，セグメントレジスタはセレクタと呼ばれる16ビット整数値を取ります．これは，メモリ上に置かれたディスクリプタ・テーブルという表へのインデックスとなる値です．実際のセグメントのアドレス値は，このディスクリプタ・テーブルに格納されており，セレクタ値に対応するディスクリプタの値と，オフセット値とで物理アドレスが合成されるわけです．

さて，このプロテクトモードでhugeポインタを実現しようとすると，どうすれば良いでしょうか？実はTurbo Cのような正規化による方式は実現が非常に困難になります．というのも，ディスクリプタ・テーブルのエントリはせいぜい8192個しかないため，正規化のたびにディスクリプタにセグメントアドレスを登録していては，ディスクリプタ・テーブルがすぐにパンクしてしまうからです．もちろん，正規化が起こるたびにディスクリプタの内容そのものを更新するという方式ならなんとか実現可能でしょうが，ディスクリプタの内容を操作する処理は非常に重いため，性能的に我慢できないものになると思われます．すなわち，プロテクトモードではMicrosoft C方式を採用しなければhugeポインタの実現はほぼ不可能なのです．逆に，Microsoft Cではプロテクトモードを念頭に置いて，コンパイラのコード生成をリアルモードとプロテクトモードで共通化するために，このようなインプリメントを採用したといえます．

Turbo C++の次期バージョンでは，Windows 3.0に対応するという噂ですが，Windows 3.0はプロテクトモードでも動作しますから，hugeポインタのインプリメントに何らかの変更が行われるのかもしれません．



ワンラインNEWS
アップルコンピュータジャパンは12月よりNIFTY Serve, 日経MIX, アスキーネット上で同社のパソコンMacintoshなどのサービス情報の提供を開始した．

hugeポインタの演算

セグメント部とオフセット部両方に行われる

hugeポインタの演算は，セグメント部とオフセット部の両方に対して行われます．たとえば，hugeポインタに整数を加算してアドレス値を変更した場合，Microsoft Cでは，まずオフセット部にその整数値が加算され，セグメントバウンダリを超えるときのみセグメント部の更新も起こります．Turbo Cでは，結果が正規化されるように処理されますから，比較的小さな値を加えてもセグメント部の更新が生じます．

hugeポインタ同士の比較は，Turbo Cでは問題なく行えます．Turbo Cのhugeポインタは演算時に常に正規化されるため，同じ物理アドレスを指すhugeポインタは同じ値をもつことが保証されるからです．

Microsoft Cの場合は，hugeポインタといえど，比較の際に正規化は行われません．つまり，同じ物理アドレスを指していても値が異なるものがありますから，やはりfarポインタの場合と同様に正規化してから比較する必要があります．

basedポインタ

MS-C 6.0から導入されたオフセットを定義できるポインタ

Microsoft C Ver. 6. 0で導入されたbasedポインタは，nearポインタと同様の16ビットポインタで，オフセットアドレスを保持します．しかし，nearポインタが通常DGROUPにおけるオフセットアドレスであるのに対して，basedポインタは何からのオフセットであるかを定義時に指定することができるものです．オフセットの「ベース」としては任意のセグメント値か，もしくは他のポインタ変数を指定します．セグメント値としては，あらかじめ定義されたセグメントや，他の変数のセグメントおよび自分自身のセグメントが指定できます．たとえば，**図17**のように記述すると，basedポインタbpは，セグメントseg1からのオフセットアドレスを保持します．_basedはbasedポインタを定義するための拡張キーワードで，常に括弧と引数をもちます．引数としては_segment型のセグメント値かポインタ変数を置きます．この_segmentも新しい拡張キーワードで，セグメント値を保持するデータ型です．セグメント値は，論理セグメント名から_segname（これも新しい拡張キーワードです）を使って生成するか，定数値などを使います．以下にbasedポインタの例をいくつか示します．

図18は，セグメント0xA000をベースとするbasedポインタです．VRAMに直接アクセスする場合などにこういう記述ができます．

図19は，farポインタfpをベースとするbasedポインタです．この場合，bpはfpの指すアドレスに対するオフセット値を保持します．すなわち，fpと同じセグメントで，fpのオフセット値にbpの値を足した値をオフセットとするアドレスを指すbasedポインタになります．

図20は，farポインタfpと同じセグメントに対するオフセットを保持するbasedポインタの宣言です．**図19**と違って，セグメントの先頭からのオフセット値になる点に注意してください．

図21は，コードセグメントをベースとしたbasedアドレスをもつ配列です．near配列やfar配列が定義できるのと同様に，based配列も定義できます．この宣言により，コードセグメント中にデータを配置するという，通常Cでは不可能なコーディングを行うことができます．

図22は，**図21**で定義したようなコードセグメント中のデータをアクセスするbasedポインタです．_segnameの引数としては，"_CODE"，"_CONST"，"_DATA"，"_STACK"の4つがあらかじめ定義されたセグメントとして使用できますが，他に自分で定義したセグメント名を与えることもできます．

図23で使われている_selfは便宜的に使用される拡張キーワードで，そのデータ自体のベースを指します．すなわち，この記述では構造体自体の置かれたセグメントに対するオフセットを保持するbasedポインタになるわけで，16ビットアドレスで連結されたコンパクトな木構造をデフォルトのデータセグメント以外で実現するものです．

basedポインタの利点は，ポインタ自体が16ビット値しかもたないのにもかかわらず，メモリ空間の任意のアドレスを指すことが可能な点です．また，ポインタの演算が16ビットで行われるため，farポインタよりも良質なコード生成を期待できます．もっとも，実際には，残念ながらそれほどコード生成の質は向上しないという報告もあります．V. 6. 0で新規にインプリメントされたばかりの機能なので，性能面の強化がまだ十分になされていないのかもしれません．

図17　basedポインタの記述例

```
char _based(seg1) *bp;
```

図18　セグメント0xA000をベースとするbasedポインタ

```
_segment vram = 0xA000;
char _based(vram) *bp;
```

図19　farポインタfpをベースとするbasedポインタ

```
char far *fp;
char _based(fp) *bp;
```

図20　farポインタfpと同じセグメントに対するオフセットを保持するbasedポインタ

```
char far *fp;
char _based((_segment)fp) *bp;
```

図21　コードセグメントをベースとしたbasedアドレスをもつ配列

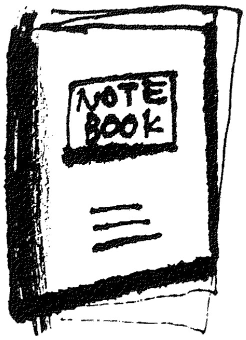
```
char _based(_segname("_CODE")) bstr[] = "string placed in CS";
```

図22　コードセグメント中のデータをアクセスするbasedポインタ

```
char _based(_segname("_CODE")) *bp;
```

図23　構造体自体の置かれたセグメントに対するオフセットを保持するbasedポインタ

```
typedef struct tree TREE;
struct tree {
    int name;
    TREE _based((_segment)_self) *left;
    TREE _based((_segment)_self) *right;
};
```



セグメントポインタ
Turbo Cのオフセット値を保持するポインタ

Turbo Cには，basedポインタに相当する機能はありません．しかし，似たような機能としてセグメントポインタというものがサポートされています．

これは，_cs，_ds，_es，_ssという拡張キーワードを使って，ポインタを修飾することで，それぞれのセグメントにおけるオフセット値を保持する16ビットポインタとするものです．たとえば，図24のように宣言すると，

これはコードセグメント中のオフセット値を保持する16ビットポインタになります．

しかし，セグメントポインタでは，比較以外の演算はfarポインタに拡張されて処理されるため，basedポインタのようにコード生成における性能改善を目的にしたものではないようです．むしろ，コードセグメントを指すポインタが欲しいといった要求に答えるためのものなのでしょう．

Turbo C++では，これに加えてセグメント値のみを保持する16ビットポインタも導入されています．これは，_segという拡張キーワードを使って，たとえば図25のように使用します．

すなわち，_segポインタにキャストすると，一般のfarポインタやfarアドレスのセグメント部分だけを取り出すことができるわけで，FP_SEGマクロとほぼ同じ効果が得られます．実際，Turbo C++のFP_SEGマクロではこのキャストが使用されています．さらに，この_segポインタに関しては，nearポインタと2項+演算子を使ってfarポインタを生成するという拡張機能もサポートされています．Turbo C++のMK_FPの定義は，この拡張機能を使って記述されています．

図24　コードセグメント中のオフセット値を保持する16ビットポインタ

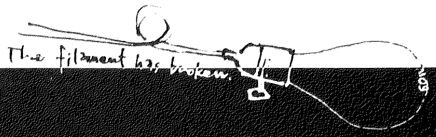
```
char _cs *cp;
```

図25　セグメント値のみを保持する16ビットポインタ

```
char _seg *sp;
char str[] = "STRING";

sp = (char _seg *)str;
```


ワンラインNEWS
アスキーは日本電気の携帯型パソコンPC-98HA（通称：ハンディ98）用のDBソフト「Technote IC card version」を発売した．



2 関数の呼び出し規約

MS-DOS上のCでは，一般に関数呼び出しは8086のCALL命令に展開されます．その際，引数はスタック上に置いて渡し，戻り値はレジスタに入れて帰すというのが一般的で，Microsoft CやTurbo Cでも基本的にはそういうインプリメントになっています．

こういった，関数呼び出しにおける引数と戻り値の受け渡しの方式を呼び出し規約（calling sequence）といいます．Microsoft CやTurbo Cは，通常のC呼び出し規約に加えて，pascal呼び出し規約というものもサポートしています．さらにMicrosoft C 6.0では_fastcall呼び出し規約というものが加されています．

C呼び出し規約
引数の個数は呼び出し側でしかわからない

printfのような可変個数の引数をもつ関数を実現するためには，関数呼び出しの引数は右から左の順にスタックにPUSHする必要があります．こうすることで，呼び出された関数は左から右の順にスタックからPOPして処理することができるからです．printfのような関数に限らず，一般にCの関数呼び出しでは，呼び出し側が与える引数の個数が，実際に関数の引数として想定された個数と一致している必要はありません．通常は，そういった引数の個数の一致しないようなコーディングはバグの元ですから行わないだけのことです．もちろん，ANSI規格で導入されたプロトタイプ宣言を行っている場合は，引数の個数の不一致はエラーとして検出されます．

このような特徴のため，Cの関数呼び出しでは，実際に与えられた引数の個数は呼び出した側でしかわからないことになります．このためC呼び出し規約では，スタック上の引数を解放するのは呼び出し側の仕事になります．関数呼び出しから復帰した直後にスタックを調整するようなコードが生成されるわけです．

pascal呼び出し規約
引数は左から右へスタックにPUSHされる

pascal呼び出し規約というのは，Microsoft Cが導入した拡張機能で，Turbo Cもこれに倣って導入したものです．この規約で宣言された関数の呼び出しでは，引数は左から右の順にスタックにPUSHされます．すなわち，printfのような可変個数の引数は実現できませんし，引数の個数が呼び出し側と呼び出される関数の側とで正確に一致していなくてはなりません．

C呼び出し規約より制約が厳しいわけですが，その反面，性能的に有利な部分もあります．関数から復帰する際には，8086のRET命令を使用するわけですが，このRET命令には，復帰する前にスタックを調整して引数を解放するという拡張された機能を持ったものがあるのです．pascal呼び出し規約の関数は，復帰する際にこの命令を使用するため，呼び出された関数側で引数を解放します．つまり，呼び出し側でスタックを調整するコードを生成する必要がないため，わずかながらコードサイズが短縮されます．実行速度の面でも，RET命令で復帰と同時に引数を解放するために若干有利になります．

_fastcall呼び出し規約
最大3個の引数をレジスタ経由で渡せる

Microsoft C 6.0では，新たに_fastcall呼び出し規約が追加されています．これは，基本的にはpascal呼び出し規約と同じで，可変個数の引数はサポートできず，呼び出し側と呼び出される関数側で引数の個数が正確に一致している必要があります．pascal呼び出し規約と違うのは，引数のうち最大3個までをレジスタ経由で渡すことができるという点です．C呼び出し規約やpascal呼び出し規約ではすべてスタック経由になるのに比べると，処理の高速化が期待できます．たとえば，**図26**のような関数の場合，3つのshort型引数はAX，DX，BXレジスタを介して渡されます．4つ以上引数が

ある場合は4つめからはスタック経由になります．またlong型の場合は1個だけで，DX:AXレジスタペアが使用されます．

Microsoft C6.0の_fastcallでは，使われるレジスタの順序は固定されているようです．つまり，左からAX，DX，BXの順序またはDX:AXペア，BXの順序となります．したがって，図27の2つの例では，前者では引数lpがDX:AX，spがBXに割当てられますが，後者ではspがAXに割当てられ，lpはスタック経由になってしまいます．

図26　_fastcallの例①

```
int _fastcall func(short a, short b, short c);
```

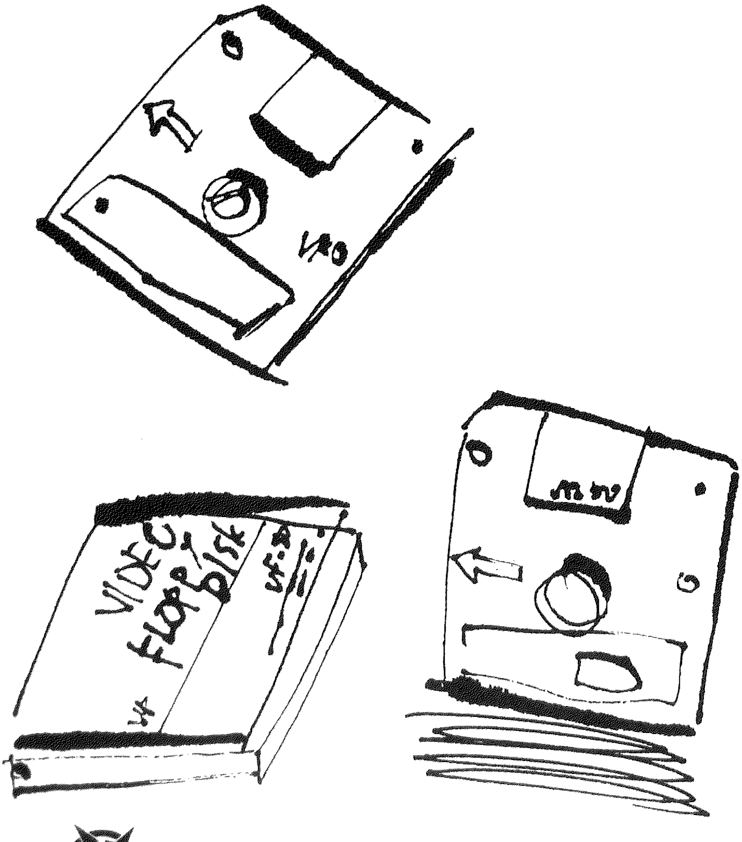
図27　_fastcallの例②

```
int _fastcall func1(long lp, short sp);
int _fastcall func2(short sp, long lp);
```

実際の_fastcall呼び出し規約の効果ですが，比較的複雑な関数に使用するとあまり速度向上の効果はないようです．なぜなら，複雑な処理を行う関数ではレジスタAX，DX，BXを通常の演算処理などに使用するために，まず入っていた引数を一度スタックに退避するコードが生成されてしまい，結果的にスタック経由で渡した場合と効率が変わらなくなってしまうからです．この場合も呼び出し側では所定のレジスタへ引数の値をセットするわけですから，むしろpascal呼び出し規約よりもすこし効率が落ちてしまうこともあります．したがってこの呼び出し規約は，どちらかというとアセンブラで記述して高速化を図る関数で使用するのに向いた機能であるといえます．

戻り値
基本的にはレジスタに戻り値を返す

一般に，関数は処理結果を戻り値として返します．この返し方は，MS-DOS上のC処理系ではレジスタ経由になるものが多いようです．Microsoft CやTurbo Cでも基本的にはレジスタに戻り値を入れて復帰します．



実際の戻り値の返しかたは，浮動小数点型を除けばMicrosoft CとTurbo Cでは同じです．すなわち，char型などの大きさ1バイトの型はALレジスタ，short型やnearポインタなどの大きさ2バイトの型はAXレジスタ，そしてlong型やfarポインタなどの大きさ4バイトの型(float型は除く)はDX:AXレジスタペアに値を入れて返します．構造体は大きさ1バイト，2バイト，4バイトのものはそれぞれAL，AX，DX:AXに返します．大きさ3バイトまたは5バイト以上の構造体は，静的なバッファに格納して，そのアドレスをAX(スモールデータモデル)またはDX:AX(ラージデータモデル)に返します．

float，doubleなどの浮動小数点型は，Microsoft Cの場合，DGROUP中の静的なバッファに格納し，そのオフセットアドレスをAXレジスタに返します．Turbo Cの場合は，コプロセッサ80x87を使用しているときはそのスタックトップレジスタに返し，エミュレータのときはエミュレータのスタックトップレジスタに相当する領域に返します．

ワンラインNEWS
米クレイコンピュータ社のスーパーコンピュータ「CRAY3」を米ローレンス・リバモア研究所から受注した．同機はこれが初受注となる．

3 インラインアセンブラ

インラインアセンブラは，Cソースプログラム中にアセンブリ言語のコードを記述する機能です．Turbo Cでは最初のバージョンからサポートされていた機能ですが，Microsoft CではVer. 6.0で初めてサポートされています．

インラインアセンブラの仕様は，だいたい両処理系ともに似たようなものですが，細かい違いがいくつかあります．Turbo Cでは，**図28**のようにasmキーワードに続けてアセンブラ命令を記述します．

このasm文は改行かセミコロンが現れるまでをアセンブラ命令と解釈しますので，セミコロンで区切れば，1行に複数のasm文を記述することも可能です（**図29**）．

したがって，asm文の行にコメントを書く場合は，アセンブラプログラムのようにセミコロンで始まるコメントは書けません．コメントはCの書式で書く必要があります．

Turbo C 2.0では，アセンブラ命令1個ごとにasmキーワードを書く必要があるため，長いコードを書くと少々面倒でした．Turbo C++では，**図30**のようにasmブロックとして複数のアセンブラ命令を一括して記述できるようになったので，比較的簡単に長いコードを書けるようになっています．

Microsoft C 6.0では，同様にインラインアセンブラを記述できますが，キーワードは _asm になります（**図31**）．

Turbo Cとは異なり，1行には_asm文は1つしか記述できないという制限があります．このため，コメントはアセンブラプログラムと同様にセミコロンで始めることが可能です．もちろん，Cの書式でも書くことができます．

どちらの処理系でも，MASMやTASMのマクロを記述することはできません．しかし，Cのプリプロセッサの処理対象には含まれますので，#defineでマクロを定義して使うことはできます．オペランドには，Cの定数，変数，ラベルなども記述することができます．

コンパイラの取扱いにも大きな違いがあります．Turbo C, Turbo C++では，インラインアセンブラが記述されたソースプログラムは，一度アセンブリ言語ソースに落としてからTASMやMASMを使ってアセンブルするという扱いになります．このため，インラインアセンブラを使用するときにはTASMかMASMが必要ですし，コンパイルに要する時間も若干長くなることになります．Microsoft C 6.0では，インラインアセンブラもCコンパイラが直接処理します．したがって，MASM等は必要ないですし，コンパイル時間が伸びることもありません．

インラインアセンブラは一般に処理の一部を高速化する目的で使用することが多いのですが，逆にコンパイラにとっては最適化をやりづらくするものになってしまいます．とくに，Microsoft C 6.0はかなり高度な最適化機能をもっていますので，インラインアセンブラを使用するとその最適化機能の大部分を殺してしまうことになります．性能改善が目的の場合は，この点をよく検討するべきでしょう．

今回で，両コンパイラのコード生成上の特徴的な部分についての話はひと通り終りました．次回からは両処理系のライブラリの特徴について比較してみたいと考えています．

図28　Turbo Cのアセンブラ記述例

```
asm mov ax,ds
```

図29　セミコロンで区切れば複数のasm文を記述できる

```
asm push es; asm mov ax,ds; asm mov es,ax
```

図30　Turbo C++のアセンブラ記述例

```
asm {
     mov  ah,13h
     sub  dx,dx
     int  18h          /* move cursor to home position */
}
```

図31　MS-C 6.0のアセンブラ記述例

```
_asm mov ax,ds
_asm {
     mov  ah,13h
     sub  dx,dx
     int  18h          ; move cursor to home position
}
```

98活用アセンブラ技法

ビデオシステム編
第3回
フォント作成プログラム

星野 操

前回はページの都合でフォントを読み込むプログラムしか掲載できませんでした．今回は，そのプログラムで使用するフォントを作成するプログラム，また読み込んだ外字を利用して枠などを描くプログラムを示します．

キャラクタのコード割り当て

キャラクタコードから形態を判別できるように

まず問題になるのは，半角ボーダーキャラクタをどのようなコードに割り当てるかという問題です．これが(IBM PCのように）わかりにくいと，実際に枠を描くのが非常にめんどうになります．いちいちキャラクタマップを参照しなければいけなくなるからです．その逆に，キャラクタコードから容易に形態を判別できるようになっていれば，プログラムを書くのも簡単になります．ここでは，IBM PCのボーダーキャラクタを参考に，以下のようにコードを割り当てることにします．

ボーダーキャラクタの決定

まず，枠線はシングルボーダー，ダブルボーダーおよびその任意の組合わせとします．ただし，キャラクタの中央でシングルラインとダブルラインがまっすぐに接続されることはないとします．縦線がダブル，横線がシングルであるような状況は許されなければいけませんが，縦線の上半分がダブル，下半分がシングルであるような状況は仮定しないということです．これはさほど不合理なことではないと思います．

PC-9801の標準的なグラフィックキャラクタでは，シングルボーダーキャラクタはひととおり用意されていますが，ダブルボーダーは横方向にしか用意されていません（しかも，シングルボーダーとダブルボーダーが交差するコーナーなどは用意されていません）．ですから，ダブルボーダーを含んだ形での完全な枠を作成することはできません．しかし，今回使用する外字キャラクタを使用すれば，任意の組合わせが可能となります．

もちろん，使用するキャラクタにグラフィックキャラクタ（半角罫線は前号に述べた理由で半角外字とは共用できない）の一部を流用すれば，必要となる外字の数は少なくなります．ただそのようにすると，当然コード体系が複雑になることから，通常は使用されない外字エリアの範囲内で自由に割り当てを行うことにします．

さて，以上の仮定のもとでコードを割り当てると，1つのボーダーキャラクタを決定するために，次の要素が決まっている必要があります．

1. 縦線がダブルかシングルか
2. 横線がダブルかシングルか
3. 縦線の上半分があるかないか
4. 縦線の下半分があるかないか
5. 横線の右半分があるかないか
6. 横線の左半分があるかないか

たとえば，縦線の上下，横線の右だけが存在すれば，「├」の字型のキャラクタになるし，上と右だけが存在すれば「L」の字型のコーナーキャラクタになります．前述の仮定がなければ，縦線のダブル・シングルの区別は上下にわけなければいけなくなるのですが，このような接続はまず考えられな

ワンラインNEWS
テクネは日本電気PC-9800シリーズ対応の1670万色フレームバッファ「FBX24/98」を2月20日発売する．

いでしょうから，属性は1つで済ませることができます．

この分け方によると，全部で64文字が必要になります．しかし，64文字全部が不可欠なわけではありません．実際，IBM PC用のボーダーキャラクタなどはもっと少ない数しか用意されていませんがそれで十分です．

縦線の上下両方の要素がない（つまり縦棒がない）場合には，縦線がダブルであるかシングルであるかを決める要素は不必要な属性になります．横線の場合でも同様です．また，このような決定の方法だと通常はあまり使用されない「キャラクタの中央まで」のキャラクタ（縦線の上半分だけがあるなど）が許容されることになります．このような「無駄」を省けば，もっと少ない文字数で表現することができます．

しかし，実際には上記の「論理コード」はきわめて合理的ですから，各種処理にはこの論理コードを使用することにします．たとえば，ラインを右にのばしていっている最中に別のラインにぶつかった場合には，そのキャラクタコードの左右線素を示すビットを設定し，描き戻せばよいことになります．

もしも，不要なキャラクタを圧縮することにするなら，実際に使用されているコードと論理コードとの間の変換・逆変換を行うテーブルを用意することが必要になります．ただし，64文字というのは，ちょうど通常使用されない外字フォントの数と同じなので，今回は前記の論理コードをそのまま（変換テーブルを介することなく）物理コードに割り当てることにしました．

もっとも，割り当て可能なコードは，JISコードで7600H～761FH，および7700H～771FHですから直接的な割り当てはできません．このため，上位バイトを1ビット右シフトし，あふれたビットを下位バイトの右から左シフトしてやります．こうすれば，簡単に論理コードからの変換を行うことができますし，その逆も可能です．

実際の論理コード

論理コードは，mkf.asmの中に記述があるように下位ビットから縦線のダブル・シングル（1でダブルボーダー），下（1で線素があり），上，横線のダブル・シングル，右，左を割り当てます．0～3Fhの6ビットで表現されることになります．これをJISコードになおす方法は前記した通りです．

縦横シングルで「├」の字型のボーダーキャラクタは，論理コードが010110B＝16Hとなります．これをJISコードに変換すると760BHとなるわけです．前号に書いたように，実際にコード7600Hおよび7700Hを使用する場合には，下位バイトの上位ビットを立てて7680Hなどにする必要があります．論理コードでいえば0および1に当たるコードの場合は特別な処理が必要になるわけですが，これらのコードはいずれも「空白」コードですから，半角スペースキャラクタ（20H）を使うことにすれば問題はないでしょう．

では，以上のコード割り当てに基づいてgaiji.exeで読み込み可能なファイルを作成するプログラムmkfについて解説しましょう．

フォント作成プログラム

64文字の半角ボーダーキャラクタを作成

プログラムの頭書きは，gaijiと同様ですから，説明は省略します．MASMを使用する場合には，疑似命令masm51は取ってください．

データセグメントに確保されているbufという変数領域は，外字一文字のフォントを格納するフォントバッファです．

その次に，フォントを作成する「関数」があります．gaijiでは実質的なサブルーチン（関数・手続き）の記述はされていませんでしたが，このプログラムではcreate_fontという関数を使用しています．

最初に，ビットマスクのための定数がequ文で定義されています．ビット単位の構造体を構成するにはほかにもいろいろな方法がありますが，ここではいちばん簡単な手を使っています．

create_font

関数を定義するには，最初にマクロ名func，それから関数名と引数を記述します．この関数では生成するキャラクタコードを引数に1つ取ることにします．その後，diレジスタを使用するので，自動的にこのレジスタが保存されるように続けてdiを〈〉で括って記述します．もちろん，siレジスタを使用する場合にはここにsi，両方ならスペースで区切って〈si di〉と記述します．

マクロfuncの内容は前回掲載しましたが，ここで「_create_font」というアセンブラのproc頭書きと「create_font」というマクロが自動的に定義されます．

まず，dsレジスタをesレジスタにコピーした後フォントバッファの内容をクリアします．これのビットを立てていくことによって，フォントを作成します．

次に，縦のラインを引きます．そのために引数コードをdxレジスタに読み込み，test命令でビットチェックを行います．proc疑似命令で引数に取られた変数は，文字マクロでword ptr形

式にキャストされています．ですから，これを読み込むためには，ワードレジスタを使うことになります．縦のキャラクタがシングルボーダーであれば，ビット00001000をパターンとして使用し，ダブルボーダーなら00010100をパターンとします（図1）．

まず，上半分のラインが必要かどうかをチェックし，必要ならrep stosb命令でパターンをセットしていきます．横のボーダーはシングルボーダーなら最上位ラインを1ラインとして8ライン目に，ダブルボーダーなら7ラインと9ラインを使用します．ですから，横がシングルボーダーであれば少なくとも8ライン目まで，ダブルボーダーであれば9ライン目まで縦線をひけばきちんと縦横のラインが接続されることになります．

次に下半分のラインをひきます．始点を固定するため，16ライン目から上に向かって線をひくことにします．上下ラインが双方使用される場合（縦ラインが貫かれている場合）には8ライン目，あるいは7ライン目から9ライン目が重ね書きされることになりますが，とくに問題はありません．

次に横ラインを書きます．横ラインはキャラクタ単位で構成されますから，縦ラインと違って，ダブルボーダーであるかシングルボーダーであるかはキャラクタパターンに関係しません．シングルボーダーである場合には8ライン目に，ダブルボーダーである場合には7と9ライン目にキャラクタパターンを書き込むだけです．そのかわり，書き込むべきパターンは縦がダブルであるかシングルであるかに依存します．

接続性を考慮すると，縦がシングルボーダーであれば左／右の線素に対応するビットパターンは11111000と00001111になりますし，ダブルボーダーの場合には1111100と00011111になります．そこで，まず縦ラインのアトリビュートにしたがって，これらのパターンをAXレジスタにセットします．

実際に横ラインを書き込むときには，diとbxレジスタをオフセットレジスタとして使用します．シングルボーダーであれば，これらを双方8ライン目（オフセット値7）に設定し，ダブルボーダーであれば7／9ライン目（オフセット値6／8）に設定し，これらのオフセット位置にパターンを書き込むことにします．つまり，シングルボーダーの場合は重ね書きをすることになるのです．左右の線素を実際に描くには，左右の線素ビットをチェックし，BX/DIのオフセットの位置にAL/AHレジスタをORすることによって行います．

さて，以上のようにラインパターンをバッファの中に作成しますが，これだけだとダブルボーダーがクロスした場合に問題が残ります．たとえば，ダブルボーダーの「+」キャラクタですと「井」状のパターンになってしまいます．これは，中央のクロス部分，すなわちダブルボーダーの間にひかれた他方の線素の一部を取り除いてやらないといけません．最後に，その調整を行います．

まずは縦横双方がダブルであった場合です．右線素があれば，7ライン目の右から3ビット目，左線素があれば右から5ビット目の垂直線素のビットはクロスを構成していますからリセットします．次に，上線素があれば7ライン目，下線素があれば9ライン目の右から4ビット目の水平線素はクロスを構成しているのでリセットします．

シングル同士のラインが交わった場合には問題は生じません．シングルとダブルでコーナーを生成した場合，またシングルのフルラインにダブルのハ

図1　mkfで作成されるフォント（8×16ドット）の構造

(A)　シングルボーダー（縦）

ライン＼ビット	1	2	3	4	5	6	7	8
1	0	0	0	0	1	0	0	0
2	0	0	0	0	1	0	0	0
3	0	0	0	0	1	0	0	0
4	0	0	0	0	1	0	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	0	1	0	0	0
7	0	0	0	0	1	0	0	0
8	0	0	0	0	1	0	0	0
9	0	0	0	0	1	0	0	0
10	0	0	0	0	1	0	0	0
11	0	0	0	0	1	0	0	0
12	0	0	0	0	1	0	0	0
13	0	0	0	0	1	0	0	0
14	0	0	0	0	1	0	0	0
15	0	0	0	0	1	0	0	0
16	0	0	0	0	1	0	0	0

(B)　シングルボーダー（横）とダブルボーダー（縦）のコーナー

ライン＼ビット	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	1	1	1	1	1
9	0	0	0	1	0	1	0	0
10	0	0	0	1	0	1	0	0
11	0	0	0	1	0	1	0	0
12	0	0	0	1	0	1	0	0
13	0	0	0	1	0	1	0	0
14	0	0	0	1	0	1	0	0
15	0	0	0	1	0	1	0	0
16	0	0	0	1	0	1	0	0

ワンラインNEWS
日経産業新聞消費研究所が調べた結果によると38%の人が自宅にパソコンを所有し，さらに日本電気がその4割を占める結果となった．

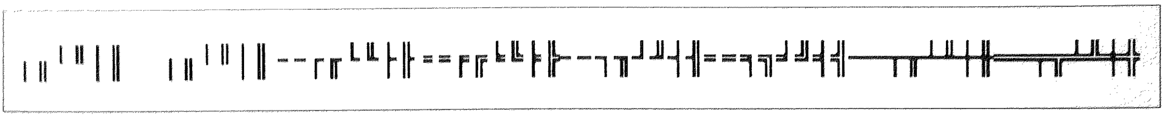


図2　mkfで作成されるフォント

ーフラインが接続されている場合も問題はありません．問題になるのは，ダブルのフルラインにシングルラインが交わっている場合です．

ダブルのフルラインとシングルのフルラインが交わっている場合に，クロスしているビットをどうするかは問題ですが，ここでは残しておくことにします（オリジナルグラフィックキャラクタの縦シングルと横ダブルの交差キャラクタと同様です）．ですから，ダブルのフルラインとシングルのハーフラインが交わっている場合にのみ処理をすることにします．

プログラムでは，まず縦ラインがダブルで横ラインがフルラインでないという条件をチェックしています．横ラインがない場合も条件を満たすのですが，リセットするのは横ラインに属するビットですから，これがもともと存在していない場合にはとくに問題は生じません．

上記条件を満たすかどうかをチェックするために，@mcマクロを使用しています．縦がダブルであり，フルラインであり，横がフルラインでない条件を@condを利用した多重条件で記述しています．これらの条件が満たされれば，縦ダブルの間に侵入している横シングルラインのビットをリセットします．逆に，横ダブル・フルラインに縦シングル・ハーフラインがつながっている場合にもビットリセットします．

以上によって指定された論理コードに対応するフォントを作成することができます（図2）．関数部分の終りは，ret文とendfマクロで示されます．

リスト1-①

```

;-----------------------------
;ボーダーキャラクタ作成プログラム
; mkf: make border char data file program
; $Log: RCS/mkf.asm $
; revision 1.2 TAN 90/10/30 05:33:26
; just modified
;
; revision 1.1 TAN 90/10/29 18:46:23
; Initial revision
;
;-----------------------------

        dosseg
        .model  small,pascal
        masm51                          ; delete if masm

        include condx.inc
        include utils.inc
        include dosfile.inc

        .stack  1000h

        .data
buf     db      16 dup (?)

        .code
;-----------------------------
;ボーダーフォント作成
; create_font: create border font
; args:
;       num: font number
;         MSB>| BL | BL | LEFT | RIGHT | D/S | UP | DOWN | D/S |<LSB

left    equ     20h
right   equ     10h
hdouble equ     8
up      equ     4
down    equ     2
vdouble equ     1

func    create_font     <num> <di>
        ; clear buffer
        push    ds
        pop     es
        mov     al,0
        mov     cx,16
        cld
        mov     di,offset buf
        rep     stosb

        mov     dx,num

        ; vertical line code set
        mov     al,8                    ; single line code
        test    dl,vdouble              ; check vertical line double/single
        @if     nz
                mov     al,14h          ; dobule line code
        @endif

        ; draw vertical line
        test    dl,up                   ; check upper half
        @if     nz
                mov     cx,8            ; for horizontal single
                test    dl,hdouble      ; check horizontal line double
                @if     nz
                        inc     cx
                @endif
                mov     di,offset buf
                rep     stosb
        @endif

        test    dl,down                 ; check lower half
        @if     nz
                mov     cx,9            ; for horizontal single
                test    dl,hdouble      ; check horizontal double
                @if     nz
                        inc     cx
                @endif
                std
                mov     di,offset buf+15
                rep     stosb
                cld
        @endif

```

エントリー部分

その次に，エントリー部分が記述されています．これはGAIJI. ASMと同様に@setupマクロを使用してから，必要な引数をスタックにプッシュし，mainを呼び出すだけです．mainでは，argc/argvを引数とした関数として記述されます．このプログラムでは引数は使用していませんが，後述するようにpascal形式での宣言をしているために引数の宣言を省略することは許されません．

fname／fd

次に，2つのローカル変数をlocal疑似命令で記述しています．fnameはフォントファイル名へのポインタとして使用し，fdはそのファイルのディスクリプタを格納します．

まず，ファイルを作成します．最初のpush/pop命令は，gaiji. asmと同様のおまじないです．ファイル名はbdfont. datであり，それを@stringマクロで作成します．アドレスはdxレジスタに入ることになるので，それをfname変数に格納します．

次に，まず書き込みモードでopenマクロを実行します．キャリーがセットされれば，ファイルが存在していないかもしれないので，そのままcreateマクロを実行します．ここでもキャリーがセットされれば書き込み禁止などのエラーの発生が考えられるので，エラーメッセージを出して終了します．メッセージの出力には@putsマクロを使用しています．

どちらかのマクロが正常に実行されれば，axレジスタにファイルディスクリプタが格納されているのでこれを変数fdに設定します．それから，単に0～63のコードに対応するフォントパ

リスト1-⑫

```
        ; horizontal line code set
        mov     ax,0f80fh               ; for vertical single
        test    dl,vdouble              ; check vertical double
        @if     nz
                mov     ax,0fc1fh       ; vertical double draw code
        @endif

        mov     bx,7                    ; for line offsets
        mov     di,bx
        test    dl,hdouble
        @if     nz
                ; double horizontal line
                dec     bx              ; modify line offsets
                inc     di
        @endif

        ; draw horizontal line
        test    dl,right                ; check right half
        @if     nz
                or      buf+bx, al
                or      buf+di, al
        @endif

        test    dl,left                 ; check left half
        @if     nz
                or      buf+bx, ah
                or      buf+di, ah
        @endif

        ; just modify connection part
        mov     al,dl
        and     al,hdouble or vdouble
        cmp     al,hdouble or vdouble
        @if     z
                ; both dobule lines
                mov     al,0ffh
                test    dl,right        ; check right line
                @if     nz
                        and     al,not 4
                @endif

                test    dl,left         ; check left line
                @if     nz
                        and     al,not 10h
                @endif

                and     buf+7,al

                test    dl,up           ; check up line
                @if     nz
                        and     buf+6,not 8
                @endif

                test    dl,down         ; check down line
                @if     nz
                        and     buf+8,not 8
                @endif
        @else
                @mc
                        test    dl,vdouble      ; check vertical double full,
                        @cond   nz
                        mov     al,dl
                        and     al,up or down
                        cmp     al,up or down
                        @cond   z
                        mov     al,dl           ; and horizontal half
                        and     al,left or right
                        cmp     al,left or right
                        @cond   nz
                @if
                        and     buf+7,not 8
                @endif

                @mc
                        test    dl,hdouble      ; check horizontal double full,
                        @cond   nz
                        mov     al,dl
                        and     al,left or right
                        cmp     al,left or right
                        @cond   z
                        mov     al,dl   ; and vertical single half
                        and     al,up or down
                        cmp     al,up or down
                        @cond   nz
                @if
                        and     buf+7,not 8
                @endif
        @endif
        ret
endf
```


ワンラインNEWS
日本電気は海外でのプリンタ生産の再編成を実施する．下位機種は全て香港で行い，欧州では各市場に適した上位機種の生産を行う．

リスト1-③

```
;--------------------------------
;初期化ルーチン
; initialize routine

entry:
        @setup
        push    _argc
        push    _argv
        call    _main
exit:
        dos     <4ch,,,>

;--------------------------------
;メインルーチン
; main routine

func    main    <argc, argv>
        local   fname, fd

        ; create file
        ; to create correct stack frame
        push    ax
        pop     ax

        @string <"bdfont.dat">
        mov     fname, dx

        open    dx,O_WRONLY
        @if     c
                create  dx
                @if     c
                        @string <"error on opening ">
                        @puts   dx
                        @puts   fname
                        mov     al,1
                        ret
                @endif
        @endif
        mov     fd,ax

        @do     si,64,0
                create_font     <si>
                write   fd,<offset buf>,16
        @loop

        close   fd

        mov     al,0
        ret
endf
        end     entry

```

ターンを，create_fontを呼び出すことによって作成し，それを順次ファイルに書き出します．gaijiのほうで7600H，7700H，7601H・・・という順にフォントを設定するようになっていますから，ここでは単に論理コード順のフォントを出力すれば済むのです．

実際にcreate_fontを呼び出すには，関数名と引数（ここではsiレジスタ）を＜＞で括って記述すれば済みます．自動的に必要なpush命令とcall命令が作成されます．定数を引数にすると，8086では使用できないコードが作成されますが，今回対象とするCPUはV30にしろ80286にしろ処理可能なコードです．

もしも引数の数があらかじめ宣言された数と異なっていると，アセンブルするときにその旨のエラーメッセージが出力されるのでわかります．PASCAL形式の呼び出しでは，呼び出された側がスタックに積まれた引数をポップアウトするので，引数のマッチングエラーは即，暴走につながります．C形式だと，直接暴走にはつながりません．ですから，このようなチェックは必須なのです．

mainは，最後にファイルをクローズし，正常終了のリターンコードを伴って復帰します．復帰後はそのまま終了のDOSコールがなされるだけです．mainはendfで括られ，最後にend疑似命令を記述します．引数には，エントリーポイントであるラベルentryを記述します．

プログラムの作成と実行

以上でフォント作成をするmkf.asmの説明は終りです．このファイルは，そのまま，

```
tasm  mkf
tlink  mkf
```

とすれば実行ファイルmkf.exeが作成されます．mkfを実行するとbdfont.datが生成されるので，前号のgaijiを実行するとこれが外字領域の下位部分にロードされます．

フォント確認プログラム

外字ボーダーキャラクタを使用して直線を描く

mkfを実行しても，このままでは本当に適切なフォントがロードされているかどうかわかりません．そこで，外字ボーダーキャラクタを使用して直線をひくためのプログラムを別に作ってみました．line.asmがそれです．

line.asmは，mainのほかに3つの関数をもっています．最初はget_addressでx/y座標からvram上のオフセットを求めるもの．次がdraw_lineで実際にラインを引く動作を行うもの．そしてatoiは引数文字列を整数に変換するものです．このうち，get_addressとatoiは関数，draw_lineは手続きということになります．

get_address

プログラムの頭書きはやはり同じです．mkfと同様な論理コードのビットパターンを認識するためのequ文が次に続いています．

それから，get_addressの定義となります．この関数は，座標x/yの2つの

引数を取り，オフセットをaxレジスタに帰します．具体的には，y＊160＋x＊2の値を計算して，それをaxに代入するだけです．PC-9801のvramは，1文字2バイトで構成されていることに注意する必要があります．

draw_line

draw_lineはこのプログラムのメイン部分です．プログラム中のコメントの通り，5つの引数を取ります．4つは2端点の座標を示すもので，最後の1つは全体をダブルボーダーで引くかシングルボーダーで引くかを決定するものです．0でシングル，非0でダブルボーダーになります．この関数ではsi/di両レジスタを使用するので，その旨の宣言があとに続きます．

この関数では，2つの端点が水平または垂直に並んでいる場合には，直線を引き，そうでない場合にはそれらを対角頂点とする四角い枠を描くことにします．ですから，まず縦または横の座標いずれかが等しいものであるかどうかを確認します．確認には，もちろん@mcマクロを使用します．

これらがともに違っていれば，枠を描きます．枠を描くには，4つの線を引けばよいわけですから，この関数自体を再帰的に使用します．関数呼び出しマクロは関数の先頭部分で定義されますから，当然，関数の内部で使用することができます．引数には，スタック内の変数引数を直接指定することができます．たとえば，siにdoubleの値を代入し，これを引数に取れば生成コードは改善されることになりますが，ここではそこまでは行っていません．4つの線を引けば，それで呼び出しは終了します．

枠でない場合には，縦または横の線を引くことになります．そこでまずx1

リスト2-①

```
;-----------------------------
;ボーダーキャラクタを使用して線を引く
; line: draw line with border char
; $Log: RCS/line.asm $
; revision 1.2 TAN 90/10/30 05:33:22
; just modified
;
; revision 1.1 TAN 90/10/29 18:46:28
; Initial revision
;
;-----------------------------

        dosseg
        .model  small,pascal
        masm51                          ; delete if masm

        include condx.inc
        include utils.inc
        include dosfile.inc

        .stack  2000h

        .code

left    equ     20h
right   equ     10h
hdouble equ     8
up      equ     4
down    equ     2
vdouble equ     1

;-----------------------------
;VRAMのオフセットアドレスを得る
; get_address: get video ram offset address
; argument:
;       x: column address
;       y: row address
; return:
;       ax: offset

func    get_address     <x, y>
        mov     ax,y                    ; 1line=160 bytes
        mov     dx,160
        mul     dx
        add     ax,x                    ; add column*2
        add     ax,x
        ret
endf

;-----------------------------
;線を引く
; draw_line: draw (rect-angle) line
; argument:
;       x1, y1: position of one edge of line
;       x2, y2: position of another edge of line
;       double: line flag
; return
;       none

func    draw_line       <x1, y1, x2, y2, double> <si di>
        ; check box/line
        @mc
                mov     ax,x1
                cmp     ax,x2
                @cond   nz
                mov     bx,y1
                cmp     bx,y2
                @cond   nz
        @if
                ; both not equall, draw box
                draw_line       <x1, y1, x1, y2, double>
                draw_line       <x1, y2, x2, y2, double>
                draw_line       <x2, y2, x2, y1, double>
                draw_line       <x2, y1, x1, y1, double>
                ret
        @endif

        cmp     ax,x2                   ; compare x1 and x2
        @if     nz
                ; horizontal line
                @if     nc
                        xchg    x2,ax   ; x1 must be smaller one
                        mov     x1,ax
                @endif

                mov     cx,0            ; setup double flag
                cmp     double,cx
                @if     nz
                        mov     cl,hdouble      ; double is yes
                @endif
                mov     ch,not hdouble  ; mask code
                or      cl,right
                mov     bx,((not right) shl 8) or left
```


ワンラインNEWS
日立製作所は4MbitDRAMのうち，チップサイズを小型化したものを今後の主力製品とする．この小型4DRAMは従来の1MbitDRAMのパッケージが流用できる．

とx2を比較し，それらが異なっている場合には横線である（つまりy1とy2が同じ）と解釈します．都合上x1＜x2であるほうが便利なので，もしもx1＞x2である場合には値の入れ替えをします．

横線を引く場合，使われるキャラクタの論理コードのフラグを引数doubleで指定されたものに直す必要があります．chレジスタはこのフラグをマスクするために使用し，clレジスタにはorすべきコードを設定するようにします．clにはまず最初にhdoubleに対応するビットを設定し，それから右線素ビットを立てます．blレジスタには左線素ビットを，bhレジスタには右線素ビットをマスクするためのパターンを設定します．diレジスタには線を引く長さ（キャラクタ数）を，siレジスタにはステップ（2）を設定します．

縦線を引く場合も同様な処理をします．まず，y1＜y2となるように，必要なら入れ替え動作を行い，次にレジスタに適切な値を設定します．上記の右線素／左線素はそれぞれ下線素・上線素に対応します．siには同様にキャラクタ数が設定されますが，diには160が設定されます．

つまり，横線を引く場合には，アドレス2をたした右隣のキャラクタに次々に線を描いていくのに対し，縦線を引く場合にはアドレス160のオフセットをもつ下のキャラクタに線を描いていくことになるからです．

以上のレジスタの設定が終わったあとは，共通のルーチンで処理をします．まず，始点である(x1, y1)の画面上のアドレスを関数get_addressを使用することによって計算します．それをdiに設定し，元のdiの値（アドレスステップ）はdxに設定します．esには，画面セグメントであるa000hを設定します．

リスト2-②

```

                mov     di,x2           ; column counter set
                sub     di,x1
                inc     di

                mov     si,2
        @else
                ; vertical line
                mov     ax,y1
                cmp     ax,y2
                @if     z
                        ret             ; null line
                @else
                @e_if   nc
                        xchg    y2,ax   ; y1 must be smaller one
                        mov     y1,ax
                @endif

                mov     cx,0            ; setup double flag
                cmp     double,cx
                @if     nz
                        mov     cl,vdouble      ; double is yes
                @endif
                mov     ch,not vdouble  ; mask code
                or      cl,down
                mov     bx,((not down) shl 8) or up

                mov     di,y2           ; row counter set
                sub     di,y1
                inc     di

                mov     si,160
        @endif

        get_address     <x1, y1>        ; get vram address
        xchg    di,ax           ; save it
        xchg    dx,ax           ; counter set
        mov     ax,0a000h       ; set vram segment
        mov     es,ax

        @begin
                ; get and decode video code
                mov     ax,es:[di]
                shr     al,1
                rcl     ah,1
                @mc
                        cmp     al,56h/2        ; gaiji high byte
                        @cond   nz
                @or
                        cmp     ah,40h
                        @cond   nc
                @if
                        mov     ax,56h/2        ; set space code
                @endif

                ; modify it
                and     ah,ch
                or      ah,cl

                ; new mask code set
                or      cl,bl
                cmp     dx,2
                @if     z
                        and     cl,bh
                @endif

                ; encode and write video code
                shr     ah,1
                rcl     al,1
                mov     es:[di],ax
                add     di,si
                dec     dx
        @until  z
        ret
endf

;-------------------------
;文字列を数値に変換する
; atoi: ascii to interger
; argument:
;       buf: pointer to string
; return:
;       ax: converted value

func    atoi    <buf>
        mov     ax,0
        mov     bx,buf
        @begin
                cmp byte ptr [bx],0
        @while  nz
                mov     dh,0
                mov     dl,[bx]
```

それから，指定キャラクタ数だけループを繰り返します．まず，diで示される画面アドレスから1ワードのキャラクタを取り出します．そして，画面コード→論理コードの変換を行います．この変換は，前に述べたようにシフトとローテートにより簡単にできます．

ここで，画面コードは上位バイトにJISコードの下位バイト，下位バイトにJISコードの上位バイト－20Hが入ったものになります．ただし，半角文字であれば上位バイトの値は0であり，下位バイトで示されるコードが有効になります．このため，JISコードの下位バイトが0であると半角文字との区別がつかなくなってしまうのです．

さて，取得した画面コードがボーダーキャラクタであるかどうかをまずチェックします．これは，画面コードの下位バイトが56H/57Hであるかどうか（つまりシフト後に56h/2になっているかどうか）と論理コードが0～3fhであるかどうかをチェックすることになります．

このために，ここでは多重条件の@orを使用しています．@mcから@orまでの@condで指定される条件，すなわちalが56h/2ではないというものか，あるいは@orから@ifまでの@condで指定される条件，すなわちahが40h以上であるというもののいずれかが満たされれば，@ifの中身が実行されるわけです．この条件が満たされる，すなわち取得したキャラクタがボーダーキャラクタでない場合には，論理コードの0を（画面コードの下位バイト指定を付けて）axレジスタに設定します．

それから，ダブル・シングルのマスクをした後clとのorを取ります．最初のキャラクタでは，右線素または下線素のみが有効になります．その後clにblとのorを取ります．これによって左線素または上線素が有効になります．2文字目からはフルラインが設定されるようになるわけです．さらにdxが2であるとき，すなわち次が最後のキャラクタである場合には，clにbhでマスクをして右線素または下線素のビットをクリアします．

以上のような手続きによって，最初のキャラクタと最後のキャラクタはハーフライン，途中のキャラクタにはフルラインの設定がなされます．もしも，

リスト2-③

```
                sub     dl,'0'
                cmp     dl,10
                @if     nc
                        ret
                @endif
                mov     cx,dx
                mov     dx,10
                mul     dx
                add     ax,cx
                inc     bx
        @repeat
        ret
endf

;------------------------------
;初期化ルーチン
; initialize routine

        .code
entry:
        @setup
        push    _argc
        push    _argv
        call    _main
exit:
        dos     <4ch,,,>

;------------------------------
;メインルーチン
; main routine

func    main    <argc, argv>
        local   x1,x2,y1,y2,mode

        ; check argument value
        cmp     argc,6
        @if     nz
                @string <'usage:line x1 y1 x2 y2 md',0dh,0ah>
                @puts   dx
                mov     al,1
                ret
        @endif

        ; parse argument
        mov     si,argv
        inc     si
        inc     si
        atoi    <[si]>
        mov     x1,ax
        inc     si
        inc     si
        atoi    <[si]>
        mov     y1,ax
        inc     si
        inc     si
        atoi    <[si]>
        mov     x2,ax
        inc     si
        inc     si
        atoi    <[si]>
        mov     y2,ax
        inc     si
        inc     si
        atoi    <[si]>
        mov     mode,ax

        ; invoke line drawing
        draw_line       <x1,y1,x2,y2,mode>

        mov     ah,1
        int     21h
        ret
endf

        end     entry

```

最初のキャラクタ位置までラインが引かれていた（たとえば左線素のみが設定されていた）場合には，右線素ビットを立てることで線が結合されることになりますし，途中のキャラクタに垂直方向のラインがすでに引かれていた場合でも問題なくコードの変換が行われることになります．

以上の処理が済んだあと，論理コード→画面コードの変換をして画面に書き戻します．ポインタdiにはオフセット値を加算し，カウンタdxはデクリメントしてこれが０になるまで処理を繰り返します．

ハーフラインのみ存在するキャラクタ

PC-9801はもちろんのことIBM PCでもハーフラインのみ存在する（たとえば，右線素のみが存在する）キャラクタは用意されていません．これは前に書いた通りです．枠を描く場合には問題になりませんが，単に線を引く場合には上記のアルゴリズムでは両端にハーフラインが残ることになります．これが好ましいかどうかはむずかしいところですが，きわめて直感的ではあると思います．

フルラインしかもっていないシステムでは，つねにその方向のフルラインを引くことになります．そして端点では，その先に線がつながっていず，かつ垂直方向の線素が存在しているときにのみハーフラインとして指定する，というようなことになります．プログラムをそのように変えるのもさほどむずかしくはないでしょう．この場合，bxレジスタにかかわる処理はまったく不要になります．

atoi関数

次のatoi関数は，Cのそれのサブセットのようなものです．アドレス引数を１つ取ります．アルゴリズムは，ポインタの指す文字が「0」～「9」であれば，axの値を10倍してから文字の数値をたすことを繰り返す，というだけの簡単なものです．

ここでは，@begin-@while-@repeatループを使用しています．@beginから@whileまでが繰り返し条件を決定するためのものであり，ポインタの示すキャラクタが文字列の終端であるかどうかをチェックしています（実は，このチェックは実質的には必要ないのですが）．これが０でないかぎり，@while-@repeatの間の部分が実行されることになります．この構造は，@beginのあとにいろいろな記述をすることによって，より複雑なループに仕立て上げることも可能です．

ループの中身では，まずポインタの指す文字が０～９の数字であるかどうかをチェックし，そうでなければそのまま呼び出し元に帰ります．有効な数字であれば，cxレジスタを利用してaxの中身を10倍し，そのあとに数値をたすことになります．それから，ポインタをインクリメントします．

この関数atoiは，mainの引数を評価するときに使用します．

コマンドラインパラメータ

そのあとにエントリーポイントの記述があり，それからmainの頭書きとなります．このプログラムでは，引数を使用しています．ローカル変数には，lineに渡す５つのパラメータに対応するものが宣言されています．

このプログラムは，コマンドラインからこれら５つのパラメータを指定するようになっています．そこで最初にargcの値をチェックし，これが６でなければusageを出力して終了するようにします．正しい数のパラメータが与えられていれば，argv配列の中身を次々にatoiの引数に取って値を変換し，それらを変数に代入します．そして最後にdraw_lineを呼び出します．

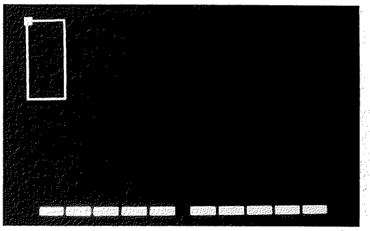
そのままだとすぐスクロールしてしまいますから，dosの一文字入力ルーチンを呼び出してキー入力を待ちます．

プログラムの作成と実行

このプログラムもmkf.asmと同様に，アセンブル・リンクするだけで実行できます．gaijiを実行したあとに

```
line 0 0 10 10 1
```

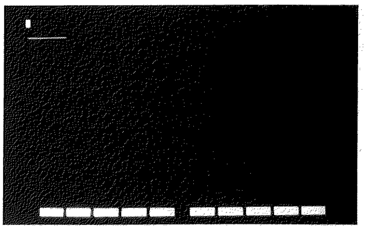
と実行すれば，ダブルボーダーで枠が書かれますし，



そのあとに

```
line 0 2 10 2 0
```

とすれば，シングルボーダーで分割線がひかれることになります．

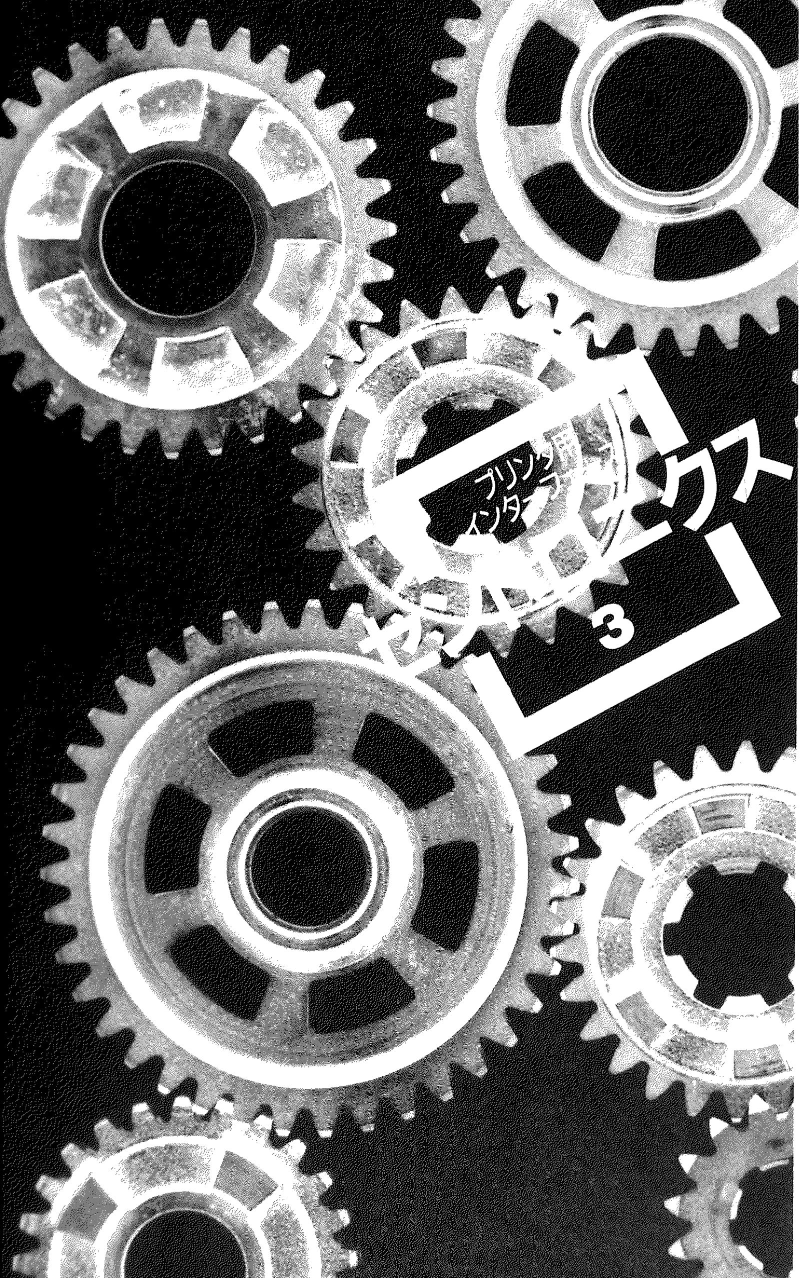


draw_lineに少し手を入れれば，そのままCなどのプログラムで呼び出せる関数になります．その仕立て方については，次回以降に述べることにします．

Reintroduction to HARDWARES

プログラマのためのハードウェア再入門

鎌田正寿



プリンタ用のインターフェースとして広く使われているセントロニクス・インターフェースは，比較的簡単に使うことができ，しかもRS-232Cなどに比べれば高速にデータが転送できるなどの特徴を持っています．ところが，このセントロニクス・インターフェースには標準規格がなく，各社さまざまな仕様で使っているため，「セントロニクス準拠」なる表現が使われているのが現状です．今回は，このセントロニクス・インターフェースについて見てみましょう．

8ビットごとにデータを転送するパラレル・インターフェース

セントロニクスは，データの転送が8ビットごとに行われるパラレル・インターフェースです．ほかにどのタイミングで信号が有効であるかを示したり，データを受け取ったことを示すハンドシェーク用の信号や，プリンタの状態を示すステータス信号などがあります．

コネクタは，通常36ピンのアンフェノールと呼ばれるものが多く使われています．プリンタ側の多くはこの36ピンのものなのですが，コンピュータ側では実装面積などの問題などによって14ピンのアンフェノール（PC-9801）や25ピンのD-SUB（IBM PCやJ-3100）なども使われますので，この場合には変換ケーブルを介して接続することになります．

では，各信号線を見てみましょう．(表1)

ピン番号は36ピン・アンフェノールで，信号の方向は「プリンタから見て」です．また信号名の最初に「-」がついているものはLowで真となる負論理，それ以外はHighで真となる正論理です．

表1　セントロニクス・インターフェースの信号線

Pin	信号名	方向	機能
1	-STROBE	入力	データを読み込むためのストローブパルス
2	DATA1	入力	データ（LSB）
3	DATA2	入力	データ
4	DATA3	入力	データ
5	DATA4	入力	データ
6	DATA5	入力	データ
7	DATA6	入力	データ
8	DATA7	入力	データ
9	DATA8	入力	データ（MSB）
10	-ACK	出力	データを受け取り，次のデータを受け取る用意ができることを示す
11	BUSY	出力	プリンタがデータを受け取れないことを示す
12	PE	出力	プリンタの紙切れを示す
13	SELECT	出力	常時HIGHレベル
14	-AUTO FEEDXT	入力	電源投入時にこの信号が真である時にはCRコードの受信によって自動的に1行の改行を行う
15			（未使用）
16	GND		信号GND
17	CHASSIS GND		プリンタシャーシの0レベル
18	NC		（未使用）
19	GND		1番ピン用のツイストペア・リターン信号GND
20	GND		2番ピン用のツイストペア・リターン信号GND
21	GND		3番ピン用のツイストペア・リターン信号GND
22	GND		4番ピン用のツイストペア・リターン信号GND
23	GND		5番ピン用のツイストペア・リターン信号GND
24	GND		6番ピン用のツイストペア・リターン信号GND
25	GND		7番ピン用のツイストペア・リターン信号GND
26	GND		8番ピン用のツイストペア・リターン信号GND
27	GND		9番ピン用のツイストペア・リターン信号GND
28	GND		10番ピン用のツイストペア・リターン信号GND
29	GND		11番ピン用のツイストペア・リターン信号GND
30	GND		12番ピン用のツイストペア・リターン信号GND
31	-INIT	入力	プリンタの初期化を行う
32	-ERROR	出力	プリンタがエラー状態にあることを示す
33	GND		信号GND
34	NC		（未使用）
35	HIGH	出力	常時HIGHレベル
36	-SLCT IN	入力	DC1/DC3を無視する

最低11本の信号線が必要

これらの信号は，標準規格が決まっていないこともあって，プリンタごとに信号の名称や使われ方もまちまちです．そのため，いろいろな設定によっても機能の有効・無効ができたりしています．また，ホストコンピュータ側からも（GNDなどは別に考えるとしても）すべての信号を制御できないのが一般的です．

プリンタを動作させるためには，最低限，GND，データ（8本），-STROBE，および-ACKまたはBUSYの片方の11本の信号線が必要となります．しかし，この-ACK/BUSYのタイミングにもさまざまなものがあります．この点について，あまり気にしなくても割と使えてしまっている現状からすると不思議にすら思われます．

では，データを送る際の信号のやり取りについて見てみることにしましょう．まず，図1を参照してください．

データは，DATA1からDATA8までの8本で送ります．このデータが有効であるのは-STROBEが真のときです．DATA1-8/-STROBE/-ACK/BUSYのそれぞれのタイミングを順に追いかけてみましょう．まず，BUSYが真ではないことを確認します．BUSYが真のときにはそれ以前のデータの処理が終わっていないなどの理由で，データを受け取ることができない状態を示しているからです．

次に，DATA1-8にデータを出力します．データを出力してからT1時間だ

け間をおいて-STROBEを真にします（この時間のことをセットアップ時間といいます）．セットアップ時間をとる理由は，それぞれの信号の通る経路・回路に必ずバラつきがでるためで，複数の信号を同時に扱う際には，このようにバラつきが安定しすべての信号が有効な状態になるまで待つ時間が必要になります．

-STROBEを真にした後に，T2時間その状態を保ちます．この時間も最小時間が決まっています．この時間の間で，プリンタ側はデータを引き取ります(データリード時間)．T2の最小時間を満足させた後で，-STROBEを真から偽の状態へ戻します．DATA1-8はまだ出力状態を保っている状態ですが，-STROBEが偽になってからT3の最小時間の間は出力状態を保たなければいけません（データホールド時間）．この時間が必要な理由は，-STROBEを偽から真へ変更するときに考慮したバラつきがありましたが，それと同じ理由で，「DATA1-8は-STROBEが真の時に有効である」という条件を満足させるためです．

ここまでで出てきたT1/T2/T3は，日電のプリンタ（PC-201H2）の場合にはそれぞれ1μ秒で，エプソンのプリンタ（HG-2550）の場合にはそれぞれ0.5μ秒です．このようにプリンタによって違いがあるので，プリンタドライバを書く場合には遅いほうに合わせ使います．

プリンタによるタイミングの違い

次に，-ACKとBUSYに注目してみます．図2を参照してください．

プリンタによってさまざまなタイミ

図1　セントロニクス・インターフェースの一般的タイムチャート

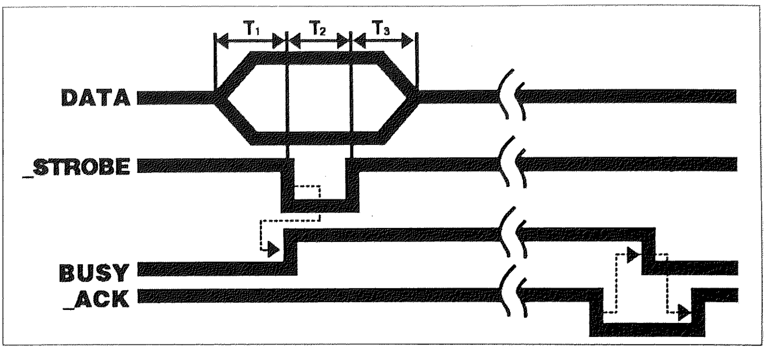


図2　プリンタの違いによるBUSY/-ACKのタイミングのいろいろ

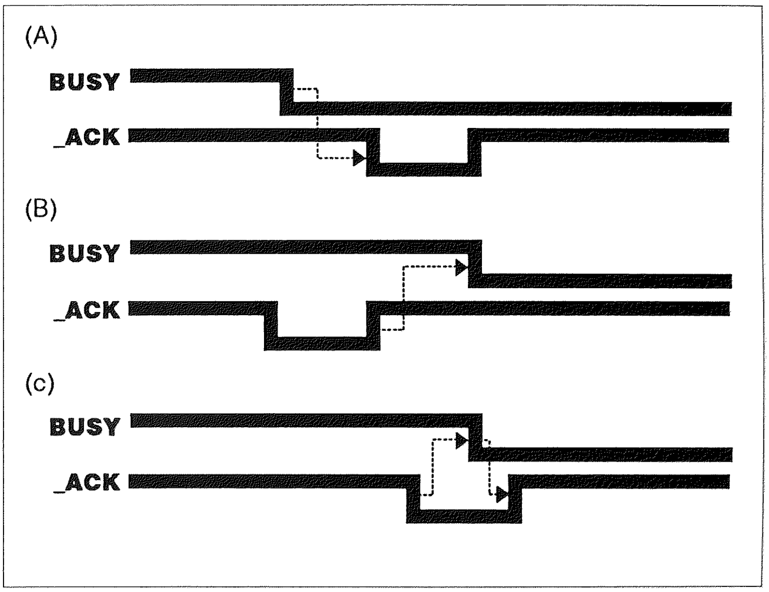
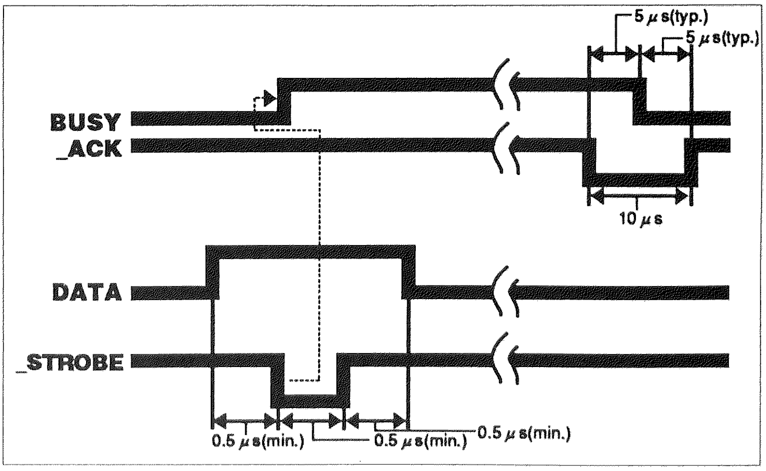
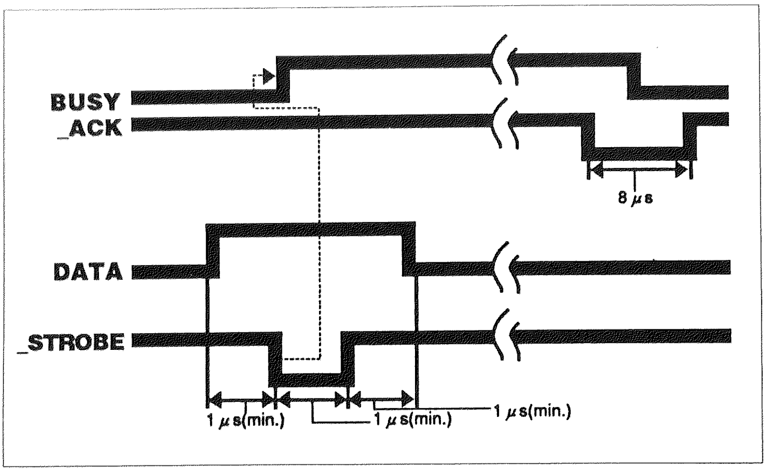


図3　エプソンHG-2550のタイムチャート



ワンラインNEWS
富士電機化学はスリランカにFDD用の磁気ヘッドの生産子会社を設立．今年3月より稼働させる．磁気ヘッドの海外生産は台湾に続いて2か所目．

図4　日本電気PC-PR201H2のタイムチャート



ングがあります．これらは，プリンタ側が出力する「データを受け取った」ことを示す-ACKと「データを受け取ることができない状態である」ことを示すBUSYのいろいろなプリンタにおけるタイミングです．コンピュータ側からドライバを書くとき，これらの信号の見方には次の3通りが考えられます．

①-ACKだけを見て次のデータを送る．

②BUSYだけを見て次のデータを送る．

③-ACKでデータ送信の完了と見なす．次のデータを送る前にBUSYを見る．

もちろん③が一番確実です．ほとんどの場合，①や②でも問題はありませんが，プリンタによってはこれでは動作しない場合もあります．

図3と図4は，2つのプリンタのタイムチャートです．

紙切れが見分けられない?

次に，PC-9801のプリンタ・インターフェースがどうなっているか見てみましょう．表2のように，GNDとデータのほかは-PSTB（-STROBE）とBUSYだけしか接続されていません．したがって，「紙切れ」や「プリンタが接続されていない」などの状態は，PC-9801では検出できないことになります．パソコンにおいては，このような仕様であっても通常プリンタはパソコンの近くに配置され，プリントアウトがされないときにその理由が操作者にとってすぐ判るような使い方がほとんどですから，これで十分なのかも知れません．

逆に遠くにプリンタが位置するような使い方をする場合には，このように簡略化された方式では不便な場合もあるはずです．たとえば，LANでつながれた遠くのプリンタサーバに出力する場合です．この場合，単にプリントアウトに時間がかかっていてそのまま待っていれば終わるのか，それとも紙切れ状態で給紙しないといけない状態なのかの違いはかなり大きいですよね．

表2　PC9801のプリンタ・インターフェース

Pin	信号名	
1	-PSTB（-STROBE）	PDB0-7が有効であることを示す
2	PDB0（DATA1）	データの最下位ビット（LSB）
3	PDB1（DATA2）	
4	PDB2（DATA3）	
5	PDB3（DATA4）	
6	PDB4（DATA5）	
7	PDB5（DATA6）	
8	PDB6（DATA7）	
9	PDB7（DATA8）	データの最上位ビット（MSB）
10	NC	（未使用）
11	BUSY	データを受け取ることのできない状態を示す
12	NC	（未使用）
13	NC	（未使用）
14	GND	信号GND

COLUMN　双方向と片方向のプリンタ・ポート

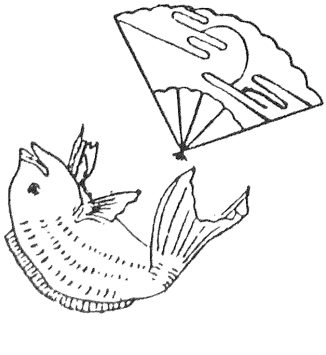
PC-9801の場合には，残念ながらプリンタポートは出力のみの片方向ですが，IBM PCやJ-3100などでは，プリンタポートは出力だけではなく入力も可能な双方向ポートになっています．

このように双方向のパラレルポートがあるといろいろな応用が考えられます．コンピュータ同士をこれで接続してファイル転送をしたり（RS-232Cより高速），イメージスキャナをつないだりもできます．人によってはここにハードディスクをつないで使っているようです．

プログラム遊工房

谷口典生

参乃巻
迷路作成プログラム



今回は，迷路を作るプログラムです．好きな大きさの迷路を画面に表示させることができます．迷路が作られていき，次に正解の道筋が浮かびあがっていく様子を見ているだけでも楽しめると思います．
このようなプログラムは，論理演算子を使うと複雑なものになってしまいますが，ここでは比較演算子を使って，できるだけ読みやすく，また冗長でなく，記述することをポイントにしました．

リスト1

```
 1: /*
 2:  * maze.c        迷路を作成、表示する
 3:  */
 4:
 5: #define TURBOC          0               /* Turbo C のときは 1 にする */
 6:
 7: #if TURBOC
 8: #include <graphics.h>
 9: #else
10: #include <graph.h>
11: #define WHITE           7
12: #define setcolor        _setcolor
13: #define moveto          _moveto
14: #define lineto          _lineto
15: #endif
16: #include <time.h>
17:
18: #define BLOCK_X 10                      /* 1区画の横のドット数 */
19: #define BLOCK_Y 10                      /* 1区画の縦のドット数 */
20: #define MAX_X   640/BLOCK_X - 1         /* 迷路の最大サイズ(横) */
21: #define MAX_Y   400/BLOCK_Y - 1         /* 迷路の最大サイズ(縦) */
22:
23: char vwall[MAX_X+1][MAX_Y];             /* 縦方向の壁 */
24: char hwall[MAX_X][MAX_Y+1];             /* 横方向の壁 */
25: char pole[MAX_X+1][MAX_Y+1];            /* 柱 */
26: int size_x, size_y;                     /* 迷路の横/縦の長さ */
27: int startx, endx;                       /* 迷路の入口/出口 */
28:
29: main(argc, argv)
30:         int argc;
31:         char *argv[];
32: {
33:         time_t t;
34:
35:         if(argc!=3) {
36:                 printf("MAZE --- 指定した大きさの迷路を作成します。¥n");
37:                 printf("usage : MAZE x y¥n");
38:                 printf("¥tx = 1..%d¥n¥ty = 1..%d¥n", MAX_X, MAX_Y);
39:                 exit(0);
40:         }
41:
42:         size_x = atoi(argv[1]);
43:         size_y = atoi(argv[2]);
44:         if(size_x<1 || size_x>MAX_X)    /* 範囲を越えていたら最大値に設定 */
45:                 size_x = MAX_X;
46:         if(size_y<1 || size_y>MAX_Y)
47:                 size_y = MAX_Y;
48:
49:         time(&t);
50:         srand((unsigned int)t);         /* 乱数の初期化 */
51:
52:         init();                         /* 画面モードの初期化 */
53:
54:         startx = rand() % size_x;       /* 迷路の入口/出口の設定 */
55:         endx = rand() % size_x;
56:         initwall();                     /* 壁データの初期化 */
57:
58:         generatemaze();                 /* 迷路を作成 */
59:         while(!kbhit());
60:         getch();
61:
62:         solvemaze();                    /* 迷路を解く */
63:         while(!kbhit());
64:         getch();
65:
66:         deinit();                       /* 画面モードの復帰 */
67: }
68:
69: initwall()                      /* 壁、柱データの初期化 */
70: {
71:         int x, y;
72:
73:         for(x=1; x<size_x; x++)
74:                 for(y=1; y<size_y; y++)
75:                         pole[x][y] = 0;
76:
77:         for(x=1; x<size_x; x++)
78:                 for(y=0; y<size_y; y++)
79:                         vwall[x][y] = 0;
80:         for(y=0; y<size_y; y++) {               /* 外壁を作る */
81:                 setvwall(0, y);
82:                 setvwall(size_x, y);
83:         }
84:
85:         for(y=1; y<size_y; y++)
86:                 for(x=0; x<size_x; x++)
87:                         hwall[x][y] = 0;
88:         for(x=0; x<size_x; x++) {
```

プログラムの使い方

このプログラムはPC-9801シリーズ用で，Turbo C, Quick CおよびMS-Cに対応しています．リスト1を打ち込み，コンパイルして実行ファイルを作ってください．Turbo Cを使う場合は6行目の「#define TURBOC 0」を「#define TURBOC 1」にしてからコンパイルしてください．

今回のプログラムではグラフィック関数を使っているので，MS-CやQuick Cでコンパイルするときにはグラフィックライブラリを指定します．MS-Cなら「CL moire.c /link graphics.lib」などとします．Turbo Cでは実行時に，BGIファイルが必要となるので注意してください．

実行するときの書式は

MAZE <x> <y>

です．<x>には迷路の横方向の大きさを，<y>には迷路の縦方向の大きさを指定します．引数を書かずに起動すると引数の範囲がわかります．デフォルトでは最大63×39の大きさの迷路まで表示できます．引数が範囲を越えた場合は最大値とみなされるので，画面いっぱいの大きさに表示するなら

MAZE 0 0

でつねに最大の迷路が作れます．もち

ろん

MAZE 40 30

とすれば40×30の大きさの迷路になります。

起動するとまずワクが描かれ，その内側が曲りくねった壁で埋められていきます．壁が作れなくなると迷路の完成です．入口と出口はそれぞれ上辺と下辺にあります．

何かキーを押すと，こんどは迷路を解きはじめます．袋小路になっている横道がつぶされ，最後に正解の道筋が浮かびあがります．さらにキーを押すと終了します．

作った迷路を残したい人のために，おまけとして迷路を出力する関数をつけておきました．リスト2のprintmaze () をmaze. cにつけ加え，main() でgeneratemaze () を呼んだ後（58行目の下）に

printmaze () ;

の1行を加えます．そして，実行するときに

MAZE 40 30 > filename

と適当なファイルにリダイレクトしてください．解答も同時に出力したいときはsolvemaze () を呼んだあとにもprintmaze () を呼ぶようにすればOKです．

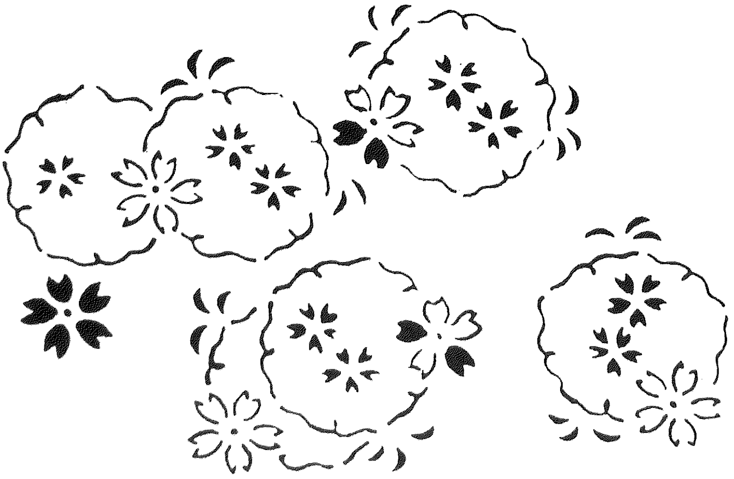
```
 89:                 if( x != startx )               /* 入口はあけておく */
 90:                         sethwall(x, 0);
 91:                 if( x != endx )                 /* 出口もあけておく */
 92:                         sethwall(x, size_y);
 93:         }
 94: }
 95:
 96: generatemaze()          /* 迷路を作成する */
 97: {
 98:         int i, x, y;
 99:
100:         for(i=0; i<(size_x*size_y/10); i++) {
101:                 while(1) {              /* 壁のある柱をランダムに選ぶ */
102:                         x = rand() % (size_x+1);
103:                         y = rand() % (size_y+1);
104:                         if(isconnect(x, y))
105:                                 break;
106:                 }
107:                 generatewall(x, y);     /* その場所から壁をのばす */
108:         }
109:
110:         for(x=0; x<size_x; x++)
111:                 for(y=0; y<size_y; y++) {
112:                         if(!isconnect(x, y))            /* 孤立した柱か? */
113:                                 sethwall(x-1, y);       /* 横壁でつなぐ */
114:                         else if (!isconnect(x+1,y) || !isconnect(x,y+1))
115:                                 generatewall(x, y);     /* 壁をのばす */
116:                 }
117: }
118:
119: generatewall(x, y)      /* (x,y)の地点からランダムに壁をのばす */
120:         int x, y;
121: {
122:         int dir, dx, dy, count;
123:
124:         for(count=0; count<10; count++) {       /* 空いている方向を10回探す */
125:                 dir = rand() % 4;               /* 壁をのばす方向 */
126:                 dx = dir-2+(dir==0)*2;          /* 方向に応じて差分を得る */
127:                 dy = dir-1-(dir==3)*2;
128:                 if(!isconnect(x+dx, y+dy)) {    /* 孤立した柱の方向なら */
129:                         count = -1;             /* カウンターを戻す */
130:                         if( dx==0 ) {
131:                                 setvwall(x, y-(dy<0));  /* 縦方向の壁を作る */
132:                                 y = y + dy;
133:                         } else {
134:                                 sethwall(x-(dx<0), y);  /* 横方向の壁を作る */
135:                                 x = x + dx;
136:                         }
137:                 }
138:         }
139: }
140:
141: isconnect(x,y)          /* 柱が孤立しているか? */
142:         int x,y;
143: {
144:         if( x<=0 || x>=size_x || y<=0 || y>=size_y )
145:                 return(1);              /* 範囲を越えていれば 1 を返す */
146:         return( pole[x][y] );
147: }
148:
149: solvemaze()             /* 迷路を解く */
150: {
151:         int x, y;
152:         int flag = 1;           /* 横道が残っている場合のフラグ */
153:
154:         while(flag) {
155:                 flag = 0;
156:                 for(x=0; x<size_x; x++)
157:                         for(y=0; y<size_y; y++) {
158:                                 if( vwall[x][y] + vwall[x+1][y] +
159:                                     hwall[x][y] + hwall[x][y+1] == 3) {
160:                                         setvwall(x, y);
161:                                         setvwall(x+1, y);
162:                                         sethwall(x, y);
163:                                         sethwall(x, y+1);
164:                                         flag = 1;
165:                                 }
166:                         }
167:         }
168: }
169:
170: setvwall(x, y)                  /* 縦方向の壁を作る */
171:         int x, y;
172: {
173:         vwall[x][y] = 1;
174:         pole[x][y] = 1;
175:         pole[x][y+1] = 1;
176:         setcolor(WHITE);
177:         moveto(x*BLOCK_X, y*BLOCK_Y);
178:         lineto(x*BLOCK_X, y*BLOCK_Y+BLOCK_Y);
179: }
180:
181: sethwall(x, y)                  /* 横方向の壁を作る */
182:         int x, y;
```

```
183: {
184:         hwall[x][y] = 1;
185:         pole[x][y] = 1;
186:         pole[x+1][y] = 1;
187:         setcolor(WHITE);
188:         moveto(x*BLOCK_X, y*BLOCK_Y);
189:         lineto(x*BLOCK_X+BLOCK_X, y*BLOCK_Y);
190: }
191:
192: #if TURBOC
193: /*
194:  * Turbo C 用の init(), deinit()
195:  */
196: init()          /* 画面モードの設定 */
197: {
198:         int graphdriver, graphmode;
199:
200:         graphdriver = DETECT;           /* 自動検出 */
201:         graphmode = PC98C8;             /* 8色モード */
202:         initgraph(&graphdriver, &graphmode, "");
203: }
204: deinit()        /* 画面モードの復帰 */
205: {
206:         restorecrtmode();               /* 画面モードを元に戻す */
207: }
208: #else
209: /*
210:  * MS-C, Quick C 用の init(), deinit()
211:  */
212: init()          /* 画面モードの設定 */
213: {
214:         _setvideomode(_98RESCOLOR);     /* グラフィック8色モード */
215: }
216: deinit()        /* 画面モードの復帰 */
217: {
218:         _setvideomode(_DEFAULTMODE);    /* デフォルト(テキスト80行)モード */
219: }
220: #endif
```

リスト2

```
printmaze()
{
        int x, y;

        for(y=0; y<=size_y; y++) {
                for(x=0; x<size_x; x++)
                        printf("%s",hwall[x][y]?"*--":"*  ");
                printf("*¥n");
                if(y<size_y)
                        for(x=0; x<=size_x; x++) {
                                printf("%s",vwall[x][y]?"|":" ");
                                if(x<size_x)
                                        printf("  ");
                        }
                printf("¥n");
        }
}
```



プログラムの説明

18-21行

各種定数を定義します.

18-19行

画面に表示する迷路の最小単位のX方向(縦方向), Y方向(横方向)のドット数を定義します. この数を小さくすればより大きい迷路を表示することができます. ただしあまり大きい迷路を作ろうとするとバッファが取れなくなるので4×4程度が限界のようです.

20-21行

上で指定した迷路の1区画のドット数と画面の横のドット数(640×400)から表示できる最大の迷路の大きさを定数に設定します.

❖main()

35-40行

引数が2個以外であれば使い方を表示して終了します. このとき引数としてとれる数の範囲を表示します. BLOCK_XとBLOCK_Yの値によって変わる迷路の最大の大きさがこれで確認できるわけです(このプログラムでは最小単位を10×10に設定しているので, 迷路の最大の大きさは63×39になります).

42-47行

第1, 第2引数をsize_x, size_yに(文字列から整数に変換して)代入し, それぞれの値がもし範囲を越えていたら最

大値に設定します.

49-50行

乱数の初期化です. これをしないといつも同じ迷路になってしまいます.

54-55行

入口と出口を迷路の上端下端のランダムな位置に設定します.

58-60行

迷路を作成する関数generatemaze ()を呼び, キー入力があるまで待ちます.

62-64行

迷路を解く関数solvemaze () を呼び, キー入力があるまで待ちます.

❖initwall()

73-75行

柱のデータpole [x] [y] を初期化します. pole [x] [y] は (x, y) 地点の柱 (縦壁と横壁の交差する場所) が壁とつながっているときに1になる変数です. 迷路の内側にある柱に0 (壁とつながっていない) を代入します.

77-83行

縦方向の壁のデータvwall [x] [y] を初期化します. vwall [x] [y] は (x, y) 地点の柱の下方にのびる壁が存在するときに1になる変数です. 内側の壁データに0を代入し, 左右の外壁を作ります.

85-93行

横方向の壁のデータhwall [x] [y] を初期化します. hwall [x] [y] は (x, y) 地点の柱の右方にのびる壁が存在するときに1になる変数です. 内側の壁データに0を代入し, 上下の外壁を作ります. このとき入口と出口にあたる部分

写真1 迷路の作成

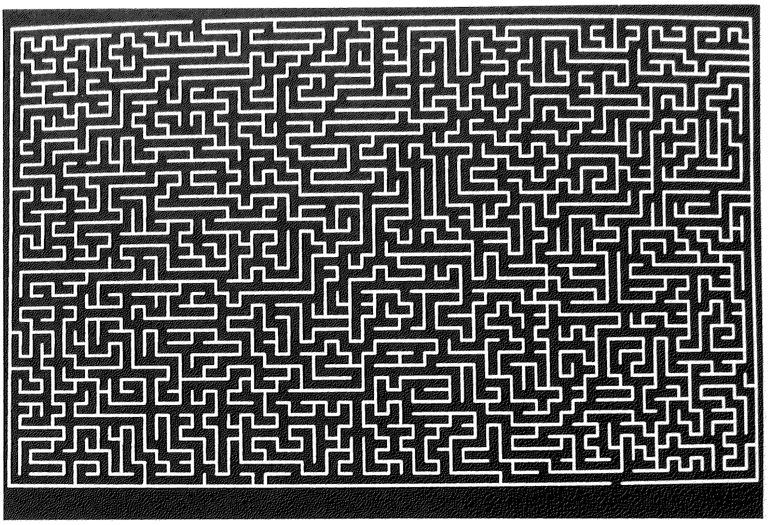
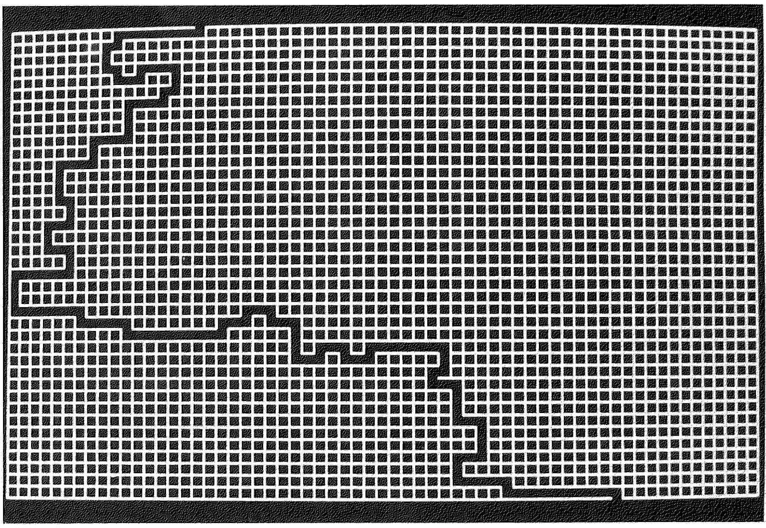


写真2 迷路を解く



だけは空けておきます.

❖generatemaze()

100-108行

適当な回数, ランダムに壁をのばしていきます. ここではその回数を（size_x×size_y/10)としていますが, できるだけ多くしたほうが質のよい迷路ができます. ただしあまり多くしても時間がかかるだけなので, 壁の生成する様子を見ていて, 壁のない広い空間がなくなる程度の回数で十分です.

始点となる柱をすでに壁のつながっている柱の中からランダムに選びます. その場所からgeneratewall () によって新しい壁をのばします.

110-116行

残った空間を埋めていきます. 左上から右下にかけて柱を調べていき, もし壁につながっていない孤立した柱があればすぐ左の柱との間に壁を作ります. 壁につながっている柱の場合は, 右と下のどちらかに孤立した柱があれば, そこからgeneratewall () を呼んで壁をのばそうとしてみます.

113行と115行のどちらか一方を削除して(「;」だけにして)実行してみると, 115行のおかげで113行が実行されることはほとんどないことがわかります. 逆に115行がないと最後に横壁ばかりが作られ, 迷路の質が落ちることがわかります.

❖generatewall()

指定された地点から空いた空間に向かってランダムに壁をのばしていきます.

124行

10回ループの形になっていますが, 129行でループ変数countの値が0に戻される場合があることに注意してください.

125行

まず壁をのばす方向を乱数で決めます. dirには0から3までの値が入り, それぞれ上左下右の方向に対応します.

126-127行

方向に対応したx方向, y方向の差分を求めます.

128-137行

その方向にある柱が孤立していれば, 実際に壁をのばす作業を行います. まずcountの値を戻す（ここでは−1を代入する）ことによって, さらに壁をのばせるようにします. x方向の差分が0なら上か下の方向なので縦方向の壁を作り, 注目する柱の座標を更新します. 同様に, 右か左の方向なら横の壁を作ります.

❖isconnect()

指定された柱が孤立した柱なら0を, 上下左右のどこかに壁があれば1を返す関数です. 範囲外の座標が指定されたときには1を返します.

❖solvemaze()

generatemaze () によって作成された迷路を解きます.

左上から右下に向かってすべての区画を調べ, 各区画について上下左右のうち3方向が壁であれば, 4方向とも壁にしてしまいます. この作業を3方向が壁である区画がなくなるまで繰り返します.

❖setvwall()

指定された座標（x，y）に縦方向の壁を作ります. vwall[x] [y]に1を代入し, 上下の柱に壁がつながったフラグを立て, 画面に壁を表示します.

❖sethwall()

指定された座標（x，y）に縦方向の壁を作ります. hwall[x] [y]に1を代入し, 上下の柱に壁がつながったフラグを立て, 画面に壁を表示します.

❖init　()

画面モードの設定を行います.

❖deinit　()

Turbo CではBGIの終了, MS-C (Quick C) では画面モードの復帰を行います.

迷路作成のアルゴリズム

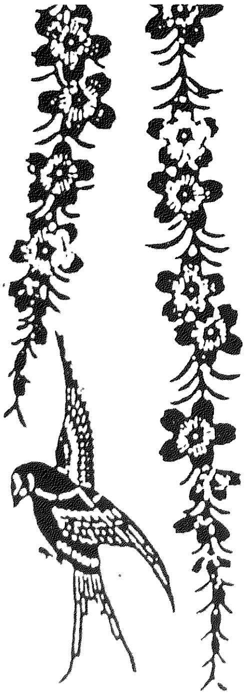
今回のプログラムで作ったのはもっともオーソドックスなタイプの迷路で, 正解の道筋がひとつしかないものです. このような迷路は, 右手（または左手）づたいに壁に沿って進んで行けば迷路のどこからでも必ず出口にたどりつけるという性質をもっています. つまり, 出口(と入口)を塞いでしまえば, 壁はすべてつながっているのです.

このことを逆に考えると, 外壁をまず作っておき, その内側に曲りくねった壁をつなげていけば迷路になるはずです. もちろん, 新しく壁を作るときは壁と壁をつなぐような場所に作ってはいけません. その部分で迷路がふたつの部分に分割されてしまうからです.

壁はできるだけ規則性のない作り方

をしなければよい迷路になりません．そのため，このプログラムでは，まず新たに壁を作る場所をランダムに選び，そこから壁を延ばしています．ある程度まで壁ができると効率が悪くなってくるので，残った空間をチェックして埋めていくという作業に入ります．具体的な方法についてはプログラムの説明のところを参照してください．迷路ができていく過程をじっくり観察したければ，適当な場所にウエイト（空ループなど）を入れてやるとよいでしょう．

迷路が完成すると，今度はその迷路を解くことになります．いろいろな方法がありますが，今回はプログラムを簡単にするため，袋小路を消していく方法を選びました．手作業で迷路を解くとき，エンピツなどで袋小路をひとつひとつ塗り潰していく人がいますが，まさにそれと同じです．ひとつひとつの区画をみていき，ちょうど3方向が壁である区画を潰す（もう一方も壁にしてしまう）ことを可能なかぎり繰り返せば，最後に入口と出口を結ぶ道だけが残るというわけです．



ワンポイントテクニック

比較演算の値を式の中で利用する

比較演算子はふたつの数（または式）の比較をする演算子のことで，等しい(==)，等しくない(!=)，より小さい(<)，より大きい(>)，等しいかより小さい(<=)，等しいかより大きい（>=）の6種類があります．

比較演算子は「if (a==2)　{････}」とか「while (b <10)　{････}」といった形で使うことが多いのですが，真で1，偽で0の値をとるという性質を使って計算式を作ると便利な場合もあります．

今回のプログラムでは，generatewall () の中でこれを利用しています（126-127，131，134行）．乱数によって生成した0から3の値に応じてdxとdyの値を決めている部分（125-127行）をswitch文によって書くと，図1のように冗長なプログラムになってしまいます．

図1　switch文を使ったプログラム

```
dir = rand() % 4;
switch( dir )
    case 0 :
        dx = 0;
        dy = -1;
        break;
    case 1 :
        dx = -1;
        dy = 0;
        break;
    case 2 :
        dx = 0;
        dy = 1;
        break;
    case 3 :
        dx = 1;
        dy = 0;
}
```

図2　少し工夫したプログラム

```
dir = rand() % 4;
dx = dir - 2;
dy = dir - 1;
if( dir==0 )
    dx = 0;
if( dir==3 )
    dy = 0;
```

図3　制御文を使わないプログラム

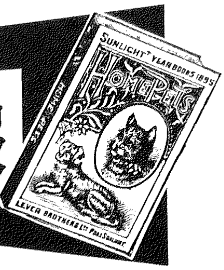
```
dir = rand() % 4;
dx = dir - 2 + (dir==0)*2;
dy = dir - 1 - (dir==3)*2;
```

ここでdir(乱数の値)とdx，dyの値の関係はdirが0，1，2，3のときdxは0，−1，0，1，dyは−1，0，1，0になるのですから，dxは(dir==0)のときをのぞいて（dir-2）という式で表され，dyは（dir==3）のときをのぞいて（dir-1）という式で表されます．これで図1は図2のように書きなおすことができることがわかります．

さらにさきほどの「比較演算子は真で1，偽で0の値をとる」という性質を利用すれば，図3のように制御文を使わずに3行のプログラムにまとめてしまうことができます．

論理演算子（&，&&，|，||，!）などを組み合わせればより複雑な式を作ることもできます．しかし，図1～3を比べてみるとわかるように，計算式を利用すれば行数は減りますが，それだけプログラムが読みづらいものになってしまいます．図1では見ただけでdx，dyがどんな値をとるかわかりますが，図3では頭の中でdirの値を0から3まで動かして計算してみないとdx，dyのとり得る値がわかりません．目で見て計算できないような複雑な式を使うよりは，場合分けも併用してプログラムの読みやすさを優先させたほうがよいでしょう．

今月の受験情報



自ら動機づけをして，まず２種から挑戦

赤松高文
東京コンピュータサービス株式会社

２種は持っていて当たり前

本号が店頭に並んでいるころには，すでに昨年10月に実施されたオンライン・特種・システム監査の各試験結果が発表され，合格者の手元には，通商産業大臣より「情報処理技術者試験合格証書」が交付されているはずです．

２種の合格者は，昨年内に発表されています．表１をご覧ください．受験率・合格率などは，ほぼ例年通りといえましょう．合格者22,222人は，昭和63年10月実施の合格者22,615人に次いで，情報処理技術者試験史上，第２番目に多い数字です．

毎年毎年，春期秋期合わせて３万人以上もが第２種に合格しています．職業プログラマとしては，「第２種は持っていて当たりマエダのクラッカー」(昭和30年代のギャグで恐縮）というわけです．

表１ 平成２年/第２回２種合格者状況

応募者数	221,235人
受験者数	142,767人
合格者数	22,222人
受験率	64.5%
合格率（対応募者）	10.0%
合格率（対受験者）	15.6%

実際，前２回のこのコーナーでは，多くの企業が情報処理試験合格者に対して，何らかの特典を与えていると述べてきました．しかし，そうした企業の中でも，第２種に関しては「合格するのはプログラマとしては当然．何しろ，小学生や70歳の素人でも合格できる試験だから」と，優遇しないところもあります．

第２種取得者に対して，月々３万円の手当を出すのが正しいのか，それとも２種は当然のこととして，待遇面でいっさい考慮しないのが正しいのか，情報処理技術者としては，前者がありがたいに決まっています．しかし，プロとしては合格して当然というのも真実でしょう．

いまから予定をたてよう

表２に今後の予定を載せておきました．第１回春期の受験申し込みはすでに終わっていますので，第１種を受験する人は，もう１年待たねばならないでしょう．表３，表４には，受験科目を示しておきましたから，参考にして今後の計画をたててください．

ただし，この原稿を書いている時点では，秋期についてはまだ決定していない段階です．昨年までと変化がないものとして記述してあります．

どの区分から受験するかは，各人の自由ですが，まだ１度も受けたことの

表２ 今後の情報処理技術者試験予定

（平成３年）	２月５日	春期願書締切
	４月21日	第１種試験 第２種試験
	６月上旬	第２種合格者発表
	６月下旬	第１種合格者発表
	７月	秋期願書受付
	10月20日頃	システム監査試験 特種試験 オンライン試験 第２種試験

（注）秋期の試験区分については未定

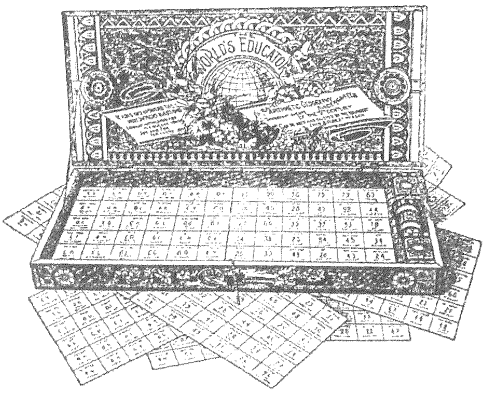
ない方は，第２種から受験していくことをお勧めします.「２種は持っていて当たり前．厚遇はしない」会社の場合では，第２種に合格しない内から，第１種・特種を受験する人が多々見受けられます．同じ受験するのなら，何らかの特典があるほうにしたい，あるいは特典のより大きいほうにするわけです．しかし，そうした目標が高いというべきか，目先の利益だけにとらわれていた人に限って，なかなか合格できないようです．第１種・特種に合格する人は，第２種から順番に合格している人が多いようです．やはり，一歩一歩地固めをしたほうが，高い目標も達せられるということでしょう．

とはいっても，「勤続疲労」がたまって過労死寸前の情報処理技術者のこと，地道に毎日勉強するのは難しいのが現実．しかし，いくら「２種は持っていて当たり前」でも勉強しなかったものは，（めったに）合格しないのも現実の姿です．ここはひとつ，受験勉強に取り組むしかないでしょう．

動機づけをしよう

プロの情報処理技術者でも，受験勉強を全くせずに情報処理技術者試験に合格するのは困難でしょう．では，いかにして勉強する気をおこすか，という動機づけが大切です．

前回でも触れたように，情報処理企業では，社員の合格率を高めるため，さまざまな努力・投資を行っています．しかしながら，会社にいろいろとやってもらわないとだめだというのは，プロとしてはどんなものでしょうか．勤



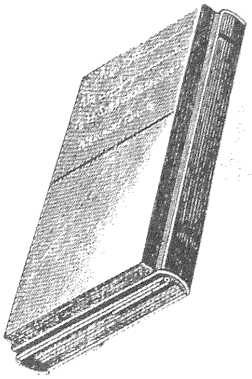


表３　秋期試験の科目・方法（予定）

試験の区分	科目	方法
システム監査 （27歳以上）	①情報処理システムの構成および機能に関する知識 ②情報処理システムの企画・開発および運用に関する知識 ③情報処理システムの監査に関する能力 ④関連知識	午前　多肢選択式（マークシート） 午後（１）　記述式 午後（２）　論述式（小論文）
特種 （25歳以上）	①ハードウェアの知識 ②ソフトウェアの知識 ③情報処理システムの設計能力 ④関連知識	午前　多肢選択式（マークシート） 午後（１）　記述式 午後（２）　論述式（小論文）
オンライン	①ハードウェアの知識 ②ソフトウェアの知識 ③オンラインシステムの設計能力 ④関連知識	午前　多肢選択式（マークシート） 午後（１）　記述式 午後（２）　記述式
第２種	①ハードウェアの知識 ②ソフトウェアの知識 ③プログラムの作成能力 ④関連知識	午前　多肢選択式（マークシート） 午後　記述式

（注）第２種で午後の「プログラム作成能力」の科目で出題されるプログラム用言語については，FORTRAN，COBOL，PL/1，アセンブラ言語のうちから選択．例年３題出題され，同じ言語だけ選択しても，あるいは何種類かの言語にわたってもかまわない．

務先からアメを与えられたり，ムチを受けたりしなくとも，自分自信でニンジンを見つけだし，奮い立ってこそプロではないでしょうか．

たとえば，きわめて個人的なことを申せば，私の場合は次の２点が第２種合格時にトリガとなりました．

まず第１点は，社内の情報処理試験対策委員になったことです．受験対策委員が合格しないことには，かっこうがつきません．委員になって半年で合格しました．

もう１点は，これはあまり参考になりませんが，受験科目の変更です．私が合格したときは，アセンブラ言語が変更される直前でした．CAP-XからCASLに変更されるとき，せっかく勉強したCAP-Xがむだにならないように１回で合格する必要があったのです．

以上２点は，適度なプレッシャーとなったと思います．人間，全くプレッシャーのない状態よりも何らかのプレッシャーがかかったほうが，より良い結果を引き出せます．あえて，自分に何らかの重圧をかけてみることをお勧めします．たとえば，会社の同じ部署

表４　プログラム用言語

FORTRAN	COBOL	PL/1	アセンブラ言語
JIS FORTRANの上位水準．ただし，いくつかの仕様は含まれない．	JIS COBOLの最高水準．ただし，いくつかの仕様は含まれない．	水準は特に規定されていない．ただし，いくつかの仕様は含まれない．	情報処理試験用の仮想CPU，COMET（16ビット）のためのアセンブラ言語CASL.仕様については，願書および試験当日の問題用紙に貼布されている．

情報処理技術者試験センター本部および支部所在地（受付時間：午前９時～午後５時15分，土日祝祭日は休業）

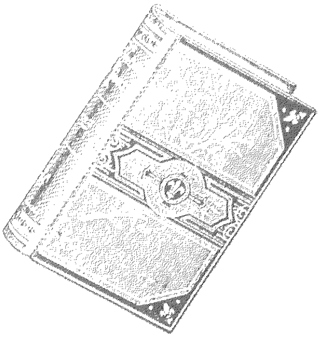
支部名	〒	所在地	電話番号	担当試験地
北海道	001	札幌市北区北24条西14-8-5 （財）北海道環境科学技術センター内	011-727-8556	札幌，帯広
東　北	980	仙台市青葉区本町3-5-3佐新ビル201号	022-227-0901	青森，盛岡，仙台，秋田，郡山
関　東	105	港区浜松町2-4-1世界貿易センタービル7F	03-3436-1321	水戸，宇都宮，前橋，埼玉，千葉，東京，八王子，横浜，厚木，新潟，長野，甲府，静岡
中　部	460	名古屋市中区栄5-26-39 タカシマ名古屋ビル8F	052-261-6818	名古屋，豊橋，富山，金沢
近　畿	540	大阪市中央区天満橋京町2-6 天満橋八千代ビル別館6F	06-946-6301	福井，京都，大阪，神戸
中　国	730	広島市中区鉄砲町1-20第3ウエノヤビル3F	082-221-4505	米子，岡山，広島，山口
四　国	760	高松市亀井町4-12セントラルビル別館7F	0878-37-2640	高松，松山，高知
九　州	812	福岡市博多区博多駅前2-9-28 福岡商工会議所ビル7F	092-472-4575	福岡，北九州，佐賀，長崎，熊本，大分，宮崎，鹿児島
沖　縄	900	那覇市久米2-2-10那覇商工会議所ビル2F	0988-62-2137	那覇
本　部	105	港区虎ノ門1-16-4アーバン虎ノ門ビル8F	03-3591-0421	テレホン・サービス 03-3591-0429

ワンラインNEWS
日本IBMは同社のコンピュータをISDNで接続するためのアダプタを発売した．同アダプタは最大1472Kbpsでデータを送信できる．

の後輩に「君のほうが先の合格したら，ホテルのディナーをおごったうえ，女の子を紹介する」と宣言してみる．すると後輩もはりきって勉強するでしょうから，一石二鳥というわけです．

もう少し積極的な動機づけとして，「転職に有利」ということがあげられるでしょう．第２種ではともかく，第１種・特種になると，いろいろと転職の誘いがかかってきます．「俺が合格するために払った努力を会社は認めようとしない」とか，「情報処理試験に合格しても，たいした恩恵がない」と思っている人は，合格してさっさと特典の多い会社に移れば良いのです．すると，情報処理合格者に逃げられた企業では，あわてて待遇を改善することになり，これまたけっこうといえましょう．もっとも，その程度の待遇面を理由に会社をコロコロかわる人間は，企業のほうでお断りかも知れません．

さて，これから合格しようという方のために，問い合わせ先を載せておきますので参考にしてください．



心当りはありませんか。

●今までに開発したプログラムを無断でコピーされた事がある。
●人に貸すプログラムをあちこちで使われたくない。
●商品ソフトの発売数に比べて、サポートの問い合わせが多すぎる。
●簡単なプロテクトがあれば試してみたい。
●ソフトを開発するにあたって違法コピーが心配だ。
●1枚しか渡していないデモ版やサンプル版がいろいろな所で使われているようだ。
●大きな需要があるソフトなのに少量しか売れていない。

違法コピー防止用自動プロテクトシステム。
AUTODISK GUARD
オートディスクガード　1セット10枚入 ¥28,000より
■実行形式ファイルを対象としたプロテクトシステムです。
■プロテクト作業は、専用のプログラムを使って簡単にできます。
■プロテクトのかかったファイルは、どこにでもコピーできますが、起動時には専用のフロッピーディスクが必要となります。
■同時に暗号化もしますので解析防止にも役立ちます。
■ハードディスクに対応したタイプもあります。
■MS-DOSが動くほとんどの機種に対応しています。

特定の人だけのセキュリティ。
PASSWORD-α
パスワードアルファ　¥28,000
■実行形式ファイルを対象としたシステムで、従来のプロテクトとは全く異質のニュータイプセキュリティソフトです。
■企業内の大切なプログラムや個人用プログラムを特定の人だけが自由に利用できるようになります。
■コマンド実行によりパスワード機能が自動的に付加されるため、誰でも簡単に利用できます。

ワンタッチで暗号化。
Asset Box
アセットボックス　¥39,800
■データファイルや文書ファイル等の不正使用を防止します。
■暗号化・パスワード等によって、正規ユーザーだけが使える環境を作ります。
■暗号化等は、メニュー画面から簡単にできます。
■暗号化したままエディトやコンパイルができます。
■ノーマルファイルとの混在も可能ですので暗号化を気にせず使えます。

サブディレクトリを丸ごとLock。
Barrier
バリアー　¥36,000
■暗号化により、解析を防止するほか、パスワード入力によってサブディレクトリィを使用できる人を制限します。
■Lockしたサブディレクトリィは隠しファイルとなり、通常のMS-DOSではアクセスできなくなります。
■オリジナルディスクを持っている人か、正しいパスワードを入力したときのみ暗号を解除でき、サブディレクトリィを使うことかできます。
■パスワードの登録はメニューを使って簡単にできます。また、サブディレクトリィのLock/Unlockは、たった一行のキー入力とパスワード入力で済みます。

対応機種（各機種共シリーズに対応しています）
●NEC：PC98●IBM：PS/2、PC AT/XT、PS/55●東芝：J-3100●富士通：FM-R・16β●日立：B16●松下：ハナコムM●AX各種

※MS-DOSは、米国マイクロソフト社の登録商標です
※記載の価格には、消費税は含まれておりません

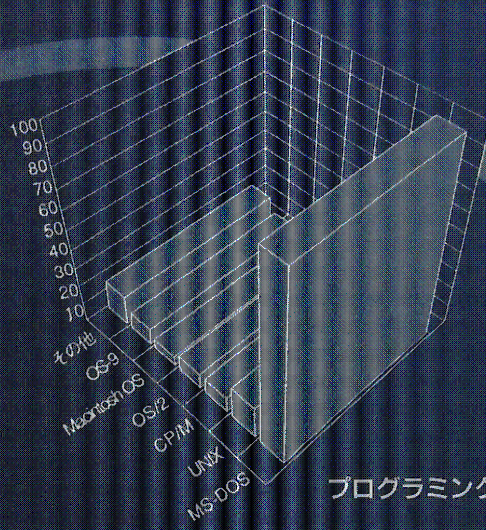
セキュリティソフト開発の専門メーカー　株式会社ランハード
〒171 東京都豊島区南池袋3-8-3 クラウンハイツ2F
TEL 03-3986-8991(代)　FAX 03-3986-8999

〈資料請求番号003〉

ワンラインNEWS
米ロータス・デベロップメント社は，同社のベストセラーソフト「1-2-3」のロシア語版を開発，ソ連向けに出荷を開始した．西側ソフトの本格的な進出は初めて．

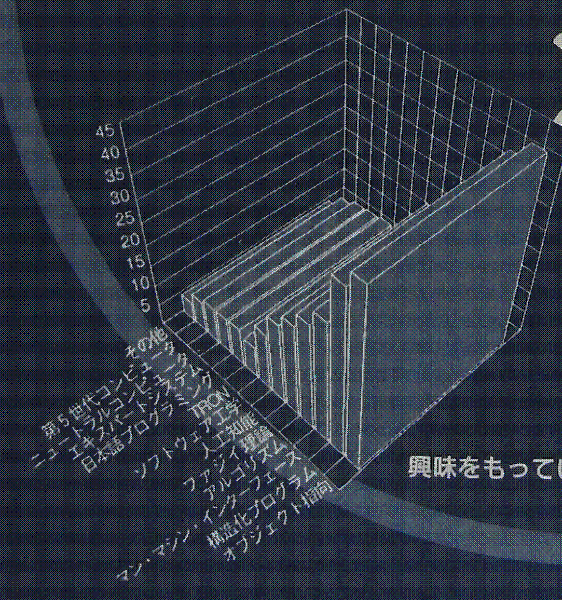
データ・ページ'91

このデータは，本誌創刊にあたって，プロ・アマチュアのプログラマを対象に実施したアンケート結果を集計したものです(有効回答者数：306名).



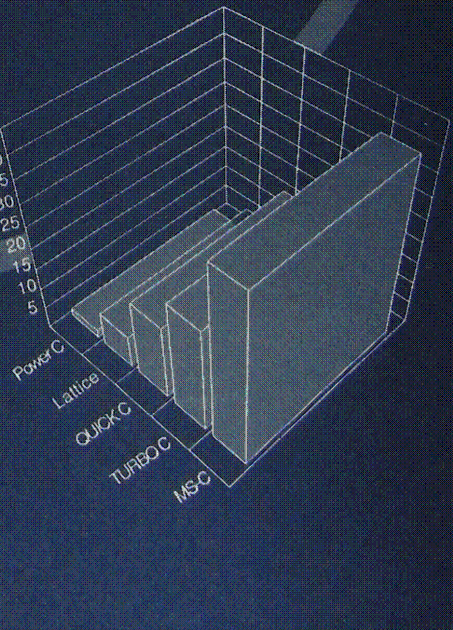
プログラミングを行っているOS

information・page 2



興味をもっているもの

使用しているコンパイラ
「C言語系」



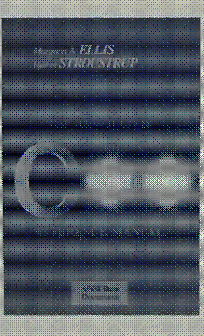
Book Review

■中村正三郎

邦訳が待たれる C++最新仕様のバイブル

■The Annotated C++ Reference Manual

Margaret A.Ellis, Bjarne Stroustrup 著
Addison Wesley



オブジェクト指向プログラミング(OOP)がポピュラーになりつつある。あまたあるOOP言語の中で広く使われるであろうと目され，現実に勢力を拡大しつつあるのが，CにOOP機能を拡張したC++である。C++は最初のクラス付きC (C with Classes) から1.0，1.1，1.2，2.0とおよそ10年間をかけて進化してきており，今後も拡張が予定されている。

本書は，C++ 2.0について，設計開発者自らが，その仕様を詳細に述べたものである。文中にはなぜこの設計を採用したか，あるいは逆に採用しなかったかのコメントも添えられている。これがAnnotatedの所以である。

Cの場合，カーニハンとリッチーの本 [1] [2] がバイブルだった。C++の場合は，以前は [3] [4] がバイブルだった。ただし [4] は訳の質が悪く [3] が手元にないと安心して読めないという難点がある。

これらに「以前は」と但し書きがつくのは，古いC++の仕様についてのバイブルだったからである。現在では，本書が最新の仕様のバイブルといってよいだろう。現状でC++の言語リファレンスを1つほしいのであれば，本書が最も薦められる選択である。

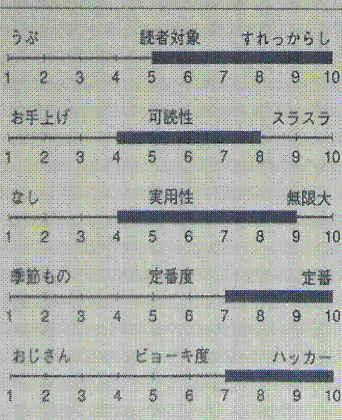
本書は通読してC++の入門とするには，向いていない。言語リファレンスだからである。したがって，C++の機能でわからなくなったときに，事典のように引いて調べる読まれ方に向いている。ちょうどCにおける [5] [6] のような趣の本である。

本書の紹介はここまでにして，C++の入門書の方に目を移してみよう。入門書といっても，以下に述べる本はいずれもC++の言語仕様の入門書であり，プログラミングの入門書ではない。かくかくしかじかのプログラミングの際に，C++ではどう書くかを述べるという意味で入門書なので，プログラミングに関する基礎知識は必要である。Cの知識があれば，なおよい。

まず，邦訳が出ている [7] [8] をお薦めする。2.0対応であるが，網羅的というよりC++の特徴的な部分に絞って解説している。いくつかのトピックに関しては十分詳しい。

一方 [9] は網羅的に解説してある。[7] [8] では扱っていないトピックも扱っている。C++の全貌について入

■評価チャート



[1] B.W.Kernighan, D.M.Ritchie著，「The C Programming Language」，Prentice Hall
[2] B.W.カーニハン，D.W.リッチー著，石田晴久訳，「プログラミング言語C」，共立出版
[3] Bjarne Stroustrup著，「The C++ Programming Language」，Addison Wesley
[4] B.ストラウストラップ著，斎藤信男訳，「プログラミング言語C++」，トッパン
[5] S.P.Harbison, G.L.Steele Jr.著，「C:A Reference Manual 2nd Edition」，Prentice Hall
[6] S.P.ハービソン，G.L.スティール著，斎藤信男監訳「詳説C言語」，日本ソフトバンク
[7] S.C.Dewhurst, K.T.Stark著，「Programming in C++」，Prentice Hall
[8] S.C.Dewhurst, K.T.Stark著，小山裕司監訳，ドキュメントシステム訳「C++言語入門」，アスキー
[9] S.B.Lippman著，「C++ Primer」，Addison Wesley
[10] 門内淳，赤堀一郎著，「C++プログラミング」，日本ソフトバンク，
[11] R.S.Wiener, L.J.Pinson著，「An Introduction to Object-Oriented Programming and C++」，Addison Wesley
[12] R.S.ウイナー，L.J.ピンソン著，前川守訳，「C++：オブジェクト指向プログラミング」，トッパン
[13] T.L.Hansen著，「The C++ Answer Book」，Addison Wesley

ワンラインNEWS
三洋電機は液晶・電池事業に積極的に投資を行うことを発表した。液晶は岐阜県に新工場を設立し，93年初めに稼働を開始する。

門的解説を1冊求めるのならば，こちらをお薦めする．邦訳も進行中と聞く．早く出てほしい．これも2.0対応である．

C++の本を求める際には，どのバージョン用かを確認した方がよい．手元の処理系と本で解説しているバージョンが違えば，本にある例がうまく動かないことがある．最近の処理系は2.0対応を謳っているものが増えているが（MS-DOSではZortech C++ 2.1, Turbo C++ 1.0などが2.0対応），古い処理系ならば注意が必要である．

古い仕様に対応したものだが，入門用として面白いものを紹介しよう．[10]は受験参考書のようなダイジェストが巧みで読みやすい．[11] [12] はOOPとはどういうものかC++を使って解説した本である．前半部はCとC++の違い，C++の言語仕様をコンパクトにまとめてある．[13] は [3] [4] にある演習問題の解答集である．多くのソースコードとその解説が掲載されている．

ともあれ，本書はLippmanの本とともに邦訳が待たれる本である．

注：Stroustrup氏と連絡を取りながら邦訳の準備を進めている人と話す機会あった．彼によれば，C++は本書出版後も仕様を拡張中であり，それが安定したらその最新仕様に基づいて邦訳を行う予定なので，今しばらく待ってほしいとのことである．

話題のネットワーク関連は，まだ洋書ばかり

古田島義和（神田・書泉グランデ）

最近，コンピュータ関連の情報誌などで，ネットワークの話題がよく目につくようになりました．その中でも，LAN Manager2.0とNetWare386が注目されており，91年早々に日本語バージョンの出荷が予定されています．しかし，この2つの出版物が和書では，『入門NetWare』（ソフトバンク）だけで，あとは洋書に頼るしかないのです．その洋書も，LAN Manager2.0の主要なものは以下の2点ぐらいです．1点は，『MS LAN Manager Programmer's Reference』（Microsoft Press）で，ネットワーキングの問題点をいかに解決するかなどを中心に解説しています．もう1点は，『LAN Manager2.0:A Developer's Guide』で，実例を使った解説書でプログラマ必見の書といえるでしょう．

一方NetWareは，『Novell NetWare386:The Complete Reference』（McGraw Hill）が，すべて網羅した1冊です．このほかに，『NetWare386 User's Guide』『The NetWare386 Manual Maker（ディスク付）』（2冊ともM&T）が出版されます．NetWareの関連図書は，現在約15点くらいが棚を埋めていますが，その中でも売れ行き良好書を数点紹介します．まず『NetWare System Interface Technical Overview』『Netware Programmer's Guide』（2冊ともAddison Wesley／以下AW）『Mastering Novell Netware』（Sybex）などがよく売れています．

先月紹介したOSF/MotifシリーズのVer1.1版が出版され，内容は次の通りです．『Programmer's Guide Relel.1』『Style Guide Relel.1』『Programmer's Reference Relel.1』『Application Environment Specification（AES） User Environment Volume』『User's Guide』．あと1点，OSFシリーズとして『Operating System Programming Interface Volume』があり，前回に続いて期待のシリーズです．

和書のベスト10では，あいかわらずX Windowが顔をだしています．6位の『オブジェクト指向入門』も注目を集めています．洋書部門の第1位の『ResEdit Reference』（AW）は，Apple社のMacintoshのリソースと呼ばれている部分を作成・変更する際に使われるビジュアルなエディタのことで，開発者向けのツールです．マニュアルは，APDA（Apple Programmer's and Developers Association）からしか購入できなかったが，書籍として発売されたおかげで，一般のユーザにも利用しやすくなりました．本書は，最新版Ver2.0dをもとに書かれています．

■和書&洋書ベスト10

（書泉グランデ調べ和書：1990年11月1日～1990年11月～30月）

順位	和書	出版社	洋書	出版社
1	XViewプログラミング入門	スペック／工学図書	ResEdit Reference	Addison Wesley
2	VZを256倍使うための本	アスキー	Compilers (paper)	Addison Wesley
3	C言語を256倍使うための本	アスキー	UNIX Network Programming	Prestice Hall
4	儀潭ソフトハウス物語	ラディク／星雲社	Macintosh Programming Primer	Addison Wesley
5	Life with UNIX	アスキー	Internetworking with TCP/IP 2/E V1	Prentice Hall
6	オブジェクト指向入門	アスキー	Macintosh C Programming Primer V2	Addison Wesley
7	プログラミング言語AWK	トッパン	Inside Smalltalk V2	Prentice Hall
8	X Window OSF/Motif プログラミング	日刊工業新聞社	Advenced X Window Applications Programming	M.I.S
9	LATEX	アスキー	Annotated C++ Reference Manual	Addison Wesley
10	たのしいUNIX	アスキー	Operating System	Prentice Hall

Programmer's Program

3種の拡張メモリに対応するメモリ管理ソフトウェア IOS-10STD

バンク&プロテクト&EMSのボードに対応

バンクメモリとはPC-9801がメモリを640KB実装していないころに拡張メモリとして登場した規格である. メインメモリの増設を行い，残ったメモリを有効に使うために考えられたもので，㈱アイ・オー・データ機器が提唱したI・Oバンク方式が一般的である.

プロテクトメモリは80286CPU上で，1MBを超えるメモリ空間にマッピングするために登場した. 通常はMS-DOSから利用できない. そこで専用のドライバを使ってアクセスする. これはオーバーヘッドが大きく，アクセス速度が遅くなる. また，80386以上のCPUでは，このメモリ空間を仮想的にEMSメモリにすることができる. 特に内部32bitバスで増設されたメモリを利用すれば，アクセス速度を期待できる.

EMSメモリはEMSボードとメモリマネージャ(EMM)を使って実現されている. 拡張ROMエリアにウィンドウを開き，16KB単位でメモリを割り当ててアクセスする. EMSメモリのハードウェアは各社独自の仕様なのでメモリマネージャも専用のを使う必要がある. EMSボードは一種のバンク方式であるためアクセス速度もプロテクトメモリよりは速い.

アイ・オー・データ機器のメモリ管理ユーティリティは，このIOS-10の3種類のボードに対応している.

拡張メモリを使うにはソフトウェアが必要

IOS-10は同社が販売するメモリボードには付属していない. 複数のメモリボードでも1本のIOS-10で管理できるので，ボード毎に付属している必要がないのである. このパッケージには各種メモリマネージャとRAMディスク，ディスクキャッシュドライバ，UBMローダ，プリントスプーラ，自動組込みプログラムなどが入っている.

IOS-10のメモリマネージャは3種類ある. EMM4J.SYSは同社のEMSボードをEMSメモリとして管理するためのメモリマネージャで，他社のメモリマネージャに比べ常駐量が小さいのが特徴である. VMM386.SYSは，80386以上のCPUを搭載したマシンで，バーチャル86モードを利用してプロテクトメモリをEMSメモリとして管理するメモリマネージャである.

最近はEMSメモリがあればデータ領域の拡大やメインメモリの占有量を減らすことができるアプリケーションも多いので，これらだけでも役に立つだろう. またこれらのメモリマネージャでUMBにRAMを割り当てるオプションを有効にすれば，デバイスやTSRを移動するUMBローダを使ってアプリケーションの使えるメモリを増大させることができる.

バンク切替えやプロテクトメモリは

従来は，バンクメモリやプロテクトメモリはアプリケーションが直接操作していたため，複数のアプリケーションから利用すると，メモリを取り合いデータが破壊される可能性があった.

これを回避するため，EMSメモリのようにメモリを管理するBMSという規格を同社は提唱している. IOS-10もBMS用のメモリマネージャBMS.SYSで，バンクメモリやプロテクトメモリを管理している. もちろん，RAMディスク・ディスクキャッシュやプリントスプーラは，EMS・BMSメモリ用のどちらも用意されている.

◆

一見，複雑なメモリ管理ではあるが，添付されている自動組込みプログラムがメモリの種類や容量を判断して，最適な環境を設定してくれる. これなら，はじめてでも迷うことなくインストール作業ができるだろう.

●IOS-10STD
販売：㈱アイ・オー・データ機器
価格：6,000円／対応機種：PC-9801シリーズおよびその互換機

ワンラインNEWS
米半導体検査装置メーカーのKLAインスツルメンツは2年後をメドに日本で生産を開始する. 同社は多摩，川崎地区を中心に東京周辺で工場用地を探している.

プログラマ登場③

VZ Editorの成功は個人プログラマの可能性を示した

兵藤嘉彦

兵藤嘉彦（ひょうどうよしひこ）
昭和37年愛知県に生まれる．名古屋工業大学電子科卒．EZ Editor，VZ Editorの開発者．

ゲームを作るために

大ヒットをとばしたVZ Editorの作者である．VZ Editorは，できるだけコンパクトに，そして速く動作するように作られたエディタで，刃物のように切れ味のいいアセンブラのプログラムだ．個人の作ったソフトが，メーカー製の製品に対抗して成功を納めたことは，パソコンプログラマに可能性を示してくれたといえる．まずは，兵藤氏のコンピュータとのつきあいについて語ってもらった．

マイコンブームのただなかに大学生になった兵藤氏は，ワンボードマイコンでコンピュータのおもしろさを知り，ベーシックマスターL3，FM7と，新しいマシンを手にいれていった．それぞれのマシンで開発用のツールなどを作ったが，ゲームに関心があり，ゲームを組むためにエディタなどのツールを作るようになったという．

大学をでると，大手メーカーのパソコン部門に入社して，ハードウェア設計に携わることになる．しかし，その会社は2年でやめた．作りたいと思っていたマシンは入社時にかなりできあがっていたし，「ハードウェアは，誰がやっても同じになるという面がある」というように，創造性を発揮する場所がなかったからである．それ以降，兵藤氏の仕事への熱意はプログラミングに向けられることになる．

フリーになって，PC-9801U2を手にいれた．それで，作ったのがVZの前身であるEZ Editorだ．このエディタは，「PCワールド」にもちこまれ販売されることになった．これは，一部で大歓迎された．つづいて，ゲームのためのグラフィックツールも作ったのだが，このときはゲームを作るまでにいたらなかった．ソフトハウスに入社してしまったからだ．そこに1年間在籍した後，兵藤氏は再びフリーの身となる．

個人の仕事は必ず残る

今度は，フリーで「食べていくために」EZ Editorを発展させたVZ Editorのプログラミングにかかった．Mifesの機能やワードスターライクなキー操作など，既存の製品を参考にしたり，人の意見を聞いたりして，機能強化に取り組んだ結果がVZ Editorである．それは，年間ソフトウェアの賞を受けるほどのヒット作となった．

「ゲームを作ろうと考えていろいろなツールを作ってきました．でも，ゲームそのものはまだ作っていないんです．今年はその形が見えるようにしたいと考えています」と，兵藤氏は話す．「いまのアプリケーションは大規模になって，組織でなければ作れないように見えますが，個人プログラマの仕事は必ず残ると思います．だから，個人でやりたいことがあるうちはフリーで……」という言葉からは，パーソナルなコンピュータ環境で活躍するプログラマの自信が感じられる．

ゲーム以外にもWindows環境，辞書など興味の範囲は広い．個人の独創性を生かした兵藤氏の作品に期待したい．

●VZ Editor

スムースな高速スクロールで話題を呼んだコストパフォーマンスの高いエディタソフト．パッケージには，PC-98版とIBM版が含まれている．兵藤氏の著作物をビレッジセンターが出版するという形になっており，VZの成功は個人プログラマの可能性を開く方法として注目される．

価格：9,800円
問い合わせ先：ビレッジ・センター
03-3239-1093

オンラインソフト採集帳 3
初心者からプロまで使える多機能&便利な通信ソフト WTERM Ver8.20

パソコン通信をはじめてみたけれど，むずしいと感じている諸氏もたくさんいると思われる．しかし，それは使っている通信ソフトに問題がある場合がほとんどである．そんな人はぜひ，今回紹介する「WTERM」を使ってチャレンジして欲しい．

まず基本設定を行いアクセス先を登録する

ハードディスクにWTERMを格納する適当なディレクトリを作り，そこへすべてのファイルを解凍する．あとはパスを通して，環境変数WTERMを設定するだけである．ここで注意することは環境変数WTERMで初期設定ファイル名まで設定すること．そうしないと起動の度にカレントディレクトリに初期設定ファイルが作られてしまう．WTERMの起動はモデムの電源が入っているのを確認した上で，

A> WTERM

と入力するだけだ．起動時のオプションはいくつかあるが，そのほとんどは起動後でも設定できる．

起動するとオープニングタイトルを表示して，ターミナルモードとなる．はじめて起動したときは，まず各種基本設定を行う必要がある．この設定は[f･7]を押し，基本設定のメニューを表示させる．設定したい項目にカーソルで反転を移動し，[RETURN]を押すとさらにウィンドウが開いて設定する内容と現在の設定が表示される．それぞれの項目を番号または[RETURN]を使って選択し，設定内容を変更をする．設定が終了したら[ESC]で１つ前の画面に戻り，同じようにほかの基本設定も行う．

つぎにアクセス先の登録である．[f･5]を押し，ダイヤルリストを表示する．登録したい場所にカーソルで反転を移動させ，[M]を押す．あとは基本設定の時と同じ要領で設定をすればいい．つぎからはアクセスしたいBBSにカーソルで反転を移動して，[RETURN]を押すだけである．

マルチダイヤルからファイル転送機能まで

WTERMの機能には，複数のネットを指定して話中の場合は順次ダイヤルする「マルチダイヤル機能」，マルチダイヤルに加えて大手商用BBSのように複数の電話番号があるネットで，グループを指定すると，そのうち１か所に接続すれば，グループの残りにはダイヤルしない「グループアクセス機能」，オンラインソフトのダウンロードに必須のバイナリファイル転送用プロトコルをほとんどサポート（子プロセスでの起動も含む）し，さらに各BBSごとにアップロードとダウンロードで使うプロトコルをそれぞれ登録して[ROLL UP][ROLL DOWN]で利用できる「ファイル転送機能」など便利な機能がある．

WTERMは多機能でありながら，操作はいたって簡単なところが大きな魅力だ．

市販ソフト顔負けのサポート体制

作者であるH.INOUE・TOMTOMの両氏はWTERMサポート用のBBSを運営している．このBBSで利用者の声に耳を傾け，それを即座にWTERMに反映させている．そして新バージョンがリリースされれば，数時間で全国のBBSに転載される．これは市販通信ソフト以上のサポート体制といっても過言ではないだろう．

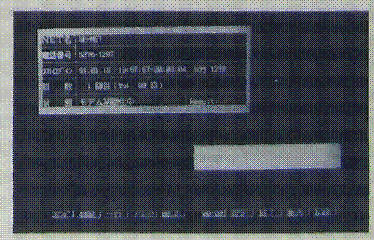
ビギナーからエキスパートユーザーまでのすべてのユーザーに薦めることができる通信ソフト，それがWTERMである．

■ **WTERM**

分類：通信ソフト／登録名：WT820.LZH（本体），WTDOC211.LZH（ドキュメント）／著作権者：H.INOUE・TOMTOM／動作環境：PC-98シリーズおよびその互換機（他にもPC-88VA・IF800・68000・J3100などの対応版あり）／掲載月日：1990年11月6日／入手月日：1990年11月6日／入手場所：WTERMサービスステーション／解凍方法：LHARC E-WT820

■ **機能および特徴**

複数のネットを指定して話中の場合は順次ダイヤルするマルチダイヤル／マルチダイヤルに加えて大手商用BBSのように複数の電話番号があるネットで，グループを指定すると，そのうち１か所に接続すれば，グループの残りにはダイヤルしないグループアクセス／夜間・深夜・休日割引きを判断して通話料金を算出する自動通話料



ワンラインNEWS
ソフト会社のコアデジタル社は神奈川県藤沢市に新しい営業・開発拠点「湘南システムプラザ」をオープンさせた．3年後50人体制を目指す．

金計算／テキスト転送（ウェイト挿入可），X-MODEM（128バイトSUM，128バイトCRC，1024バイトCRC，各フライング），Y-MODEM（G，BATCH）そしてMLINKを本体にもち，Z-MODEMやQuickVAN，B-Pulsなどを登録して面倒な子プロセスでの起動を自動的に行えるファイル転送機能／各ネット毎に利用するファイル転送プロトコルをアップ・ダウン別に設定し，ワンキーで起動可能／テキストのアップロード時改行コード挿入機能／強力なマクロ言語／エスケープシーケンスにも対応したバックスクロール，バッファは最大64KBで，表・裏のグラフィックV-RAMをバッファとして指定可能／履歴バッファ付き子プロセスの起動／履歴画面の切取&送出／各種エディタの呼出し／圧縮ファイルの解凍，閲覧／ラジオパケットに対応／シフトJIS，新JIS，旧JIS，EUCの各種漢字コード対応（受信は自動判別）／行末キャラクター，DELコードの設定／ビット長，パリティ指定，ストップビット長の設定

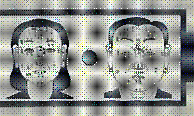
ファイル構成

<WT820.LZH>

Name	Original	Packed	Ratio	Date	Time	Attr	Type	CRC
READ.ME	13921	6475	46.5%	90-11-06	0:00:00	a--w	-lh1-	E295
WTERM.EXE	178400	105717	59.3%	90-11-06	0:00:00	a--w	-lh1-	F063
2 files	192321	112192	58.3%	90-11-06	0:00:00			

<WTDOC211.LZH>

Name	Original	Packed	Ratio	Date	Time	Attr	Type	CRC
WMACRO.DOC	18929	6732	35.6%	90-11-06	0:00:00	a--w	-lh1-	F341
WTERM.DOC	141394	52367	37.0%	90-11-06	0:00:00	a--w	-lh1-	217E
WWIVMAC.LZH	3992	3867	96.9%	90-03-20	0:00:00	a--w	-lh1-	343F
3 files	164315	62966	38.3%	90-11-06	0:00:00			

HEALTHY?  UNHEALTHY?

③ ポカポカ，スベスベ 入浴剤は効き目？気分？

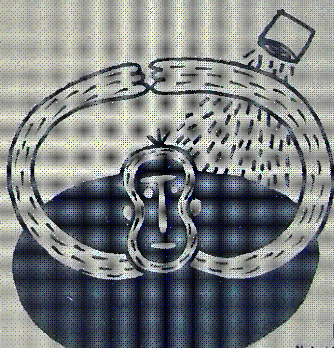


Illustration
Nobuki Fukuhara

冬はやっぱりお風呂がいちばん．というわけでもないが，入浴剤に凝りはじめると，毎日「今日はみちのくの湯で温泉気分」「ミルキーローズでお肌スベスベ」など，バスタイムにリッチな気分を味わえる．以下に売れ筋のものをとりあげてみたので，今年の冬はひととおり試して，お風呂ツウになることをお薦めする．

■温泉シリーズ

①日本の名湯（加賀の湯山代，みちのくの湯作並）

②旅の宿（十和田，霧島，奥飛騨）

温泉と同じ成分を含む．老若男女に根強い人気

■薬効成分を含んだもの

①まんのう・湯，シャンラブ，健妙泉

にんにくB1エキス入り．アトピー性皮膚炎に効く．

②クナイプ

西独のクアハウス（温泉保養施設）で使用されている薬草が主成分．お風呂ツウに受けている．

■美容に効果があるもの

①ロモコート浴剤（米ぬか油）

②ニューヨークタイムズ固形（ミルキーオイル）

主成分中のオイルが肌をスベスベに．女性に人気

■発泡・カラー・香りなど

①トリム（ブクブク発泡）

②かいじゅう風呂（ベジタブルの色と香り）

③バスクリン，おふろどんと，バブ（一般的な家庭用）

コンピュータの法律

プログラム著作権とはどんなものか——その 2

◆

熊谷信太郎(弁護士)

プログラムに代表されるコンピュータソフトは，著作権法により法的保護が与えられています．前回は著作権に関する基本的な説明をしましたが，今回はプログラムと著作権に関する具体的な問題を検討したいと思います．

■©マークと著作権

プログラムの作成者は，作成と同時に，特別な手続きを要することなく，当然に著作権を取得します．これを，いわゆる無方式主義といいます*1．市販のプログラムには©マークをつけたものが多いので，©マークをつけないと著作権の保護を受けられないと勘違いをしている人が多いのですが，日本は無方式主義を採用していますから，プログラム作成者が©マークを付けなくてもプログラム著作権は発生します．

もともと，©マークの表示は，著作権法の保護を受けるために一定の方式を要求する方式主義を採用している国において，著作権の保護を受けるために必要とされたものです．かつて，アメリカは方式主義をとっていましたから，アメリカにプログラムを輸出するような場合には，©マークをつける必要がありました．しかし，その後アメリカにおいても，©マークの表示をしなくても著作権の保護を受けられるような制度に変わっています．

■著作権が認められないプログラム

プログラムは，著作権法で著作物として保護されていますが，実はすべてのプログラムが著作権法の保護を受けるわけではありません．

著作物として著作権法の保護を受けるためには，「創作的表現」がなされたものでなければならず*2，「創作性」を欠くプログラムは著作権法の保護を受けられないのです．

もっとも，従来は，この「創作性」はそれほど高度の水準のものである必要はなく，また，著作物としての価値(文学的，芸術的価値など)が低くても，創作性が否定されるものではないとされてきました．

ところが，プログラムは，実用的な目的のもとに使用される，技術的な創作物ですから，「創作性」も，文芸，美術などに比べた場合，より高度の水準が要求されるべきだと考えられています．

そのため，すべてのプログラムが著作権法の保護を受けるのではなく，たとえば，きわめて少数の命令よりなる，誰が使っても同じような短いプログラムは技術的な「創作性」がなく，したがって，著作権は認められないのです．

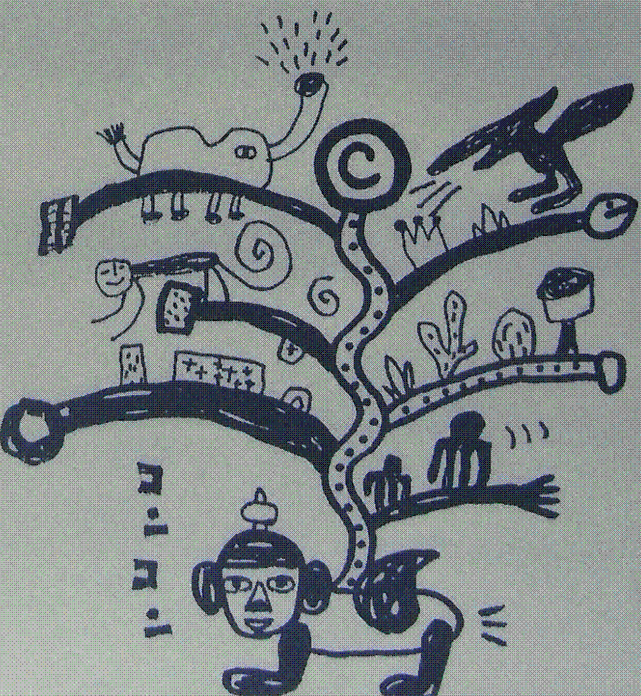


Illustration : Nobuki Fukuhara

ワンラインNEWS
富士通は低速データを多重化し，高速デジタル専用線で送信するための多重化装置「CompactMUX」を開発した．6回線分をまとめることができる．

■バグのあるプログラムと著作権

バグがあるために，そのままでは作動しないプログラムにも，著作権が認められます．バグがあるプログラムは著作物としての価値は低いのですが，だからといって，「創造性」が否定されるものではないからです．また，もしこれに著作権を認めないと，プログラム作成者と別の者がバグを直してプログラムが作動するようにした場合，その者にプログラム全体の著作権が発生するというおかしな結果になってしまうからです．

これと同様のことは，未完成のプログラムについてもいえるでしょう．

もっとも，いずれの場合でも，プログラムといえる程度のまとまりを持っている必要があります．バグの程度がひどすぎてプログラムとはいえないようなものや，未完成というより，プログラムのごく一部しかできていないというようなものは，まだ，プログラム著作物として保護する実体を備えていないというべきでしょう．

＊1　著作権法第17条第2項
＊2　著作権法第2条第1号：著作物とは，思想又は感情を創作的に表現したものであって，文芸，学術，美術又は音楽の範囲に属するものをいう．

アキバ短信

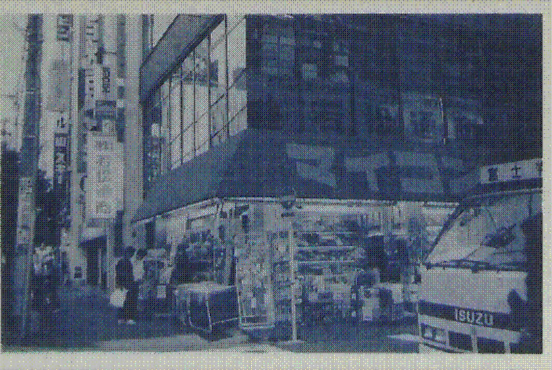
営業時間　10:00～19:30　年中無休
TEL　03-3251-4121

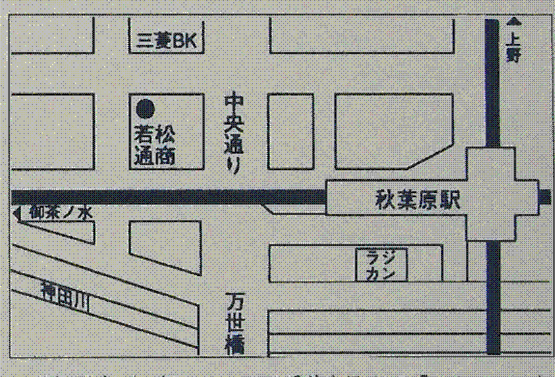
若松通商

店名からは何の店かわからないアキバらしさ

「名は体を表す」という言葉は秋葉原電気街では通用しない．「サトームセン」には家電製品が並び「ラジオ会館」はパソコンとオーディオの専門店ばかり．「〇〇電機」や「××デンキ」が多いのはともかく「△△無線」「□□商会」にいたっては，何を売る店なのか想像もつかない．

若松通商というのもわからない名前の1つである．店先にはCD，電子手帳，時計，データ切替スイッチなどが並んでいるのだが，店内に一歩足を踏み入れるとそこは各種パーツやICがズラリ．2階も電子デバイス専門のパーツショップで，階段を上がっていくと壁にはさまざまなICの価格がひとつずつ紙に書いて貼りだしてある．3～6階には他の店やショールームが入っているのだが，7階にはまたしても各種センサー専門のパーツショップ．他の階とは少々毛色が違うのが地下1階．こちらはパソコン，ワープロ，ハードディスク，ボード類が中心だ．





若松通商（一部のマニアは愛着を込めて「ワカマツ」と呼ぶ）は名前や外見では予断を許さないきわめてアキバ的な構造を持った店だといえるだろう．こういう店は一般のパソコンユーザーや日曜プログラマにはまず縁がない．普通なら店を訪れる機会も，その名を聞くことすらない（はずだった）．ところが最近ワカマツはちょっとばかり有名なのである．

ことはあるBBSにダイナブックをスピードアップする方法というのがアップされたことにはじまる．要は内蔵のクロックICを交換して本来10MHzで動作するものを12MHzで動かそうというのであるが，作者は「24MHzのクロックICを秋葉原の若松通商で買いました」と記しているのだそうだ．かくして若松通商の名はダイナブックユーザーの間に知れわたったという．

Vertual Space Guide

Ilusration : TOSHIYA SHIROTANI

バーチャルスペースガイド

きっちり作られた仮想世界

C平　やっと出ましたね，2年2カ月ぶり．

P子　何の話？まさか「チビまる子ちゃん」の第7巻じゃあないよね．

C平　ちがうちがう，「AKIRA」第5巻の話．これで終わりかと思ったらまだまだなんだ．何巻まで続くんだろうね，連載も読んだり，アニメのLDももっているけれど，あのムックみたいな単行本が，大きさもまとまりもよくていちばんおもしろく読める．

P子　そうだよね．絵は書き込んであるから迫力あるけど，ストーリーはなかなか進まないから連載だとつらいものがあるよ．

C平　AKIRAとか，大友克洋の作品の多くは，分類してしまうと超能力SFマンガってことになるんだろうけど，小説・マンガを含めて超能力やオカルトものに見られるセコさがない．UFOものなんかによくある「地球は狙われているんだぞ」みないなこけおどしにはうんざりだものね．そこへいくとAKIRAは仮想世界がきっちり作られていると感じる．

P子　そうだよね．超能力という形をとっているけれど，物質と精神ってやつ，人間の抱えている問題っていうのかな．そういったテーマにリアリティが感じられるもん．でも，私はこけおどしみないなのも好きだよ．なんか，わけのわからないエネルギーが感じられるものがあるから．楳図かずおなんかね．

C平　でも，エネルギーだけだと疲れてしまうな．

P子　それはそうかもね．ソフトでもタイトル画面ばっか，グラフィック使ってチャラチャラやってるやつは，使っているうちにうるさく感じてしまうもの．どうせグラフィック使うならWindows 3.0みたいにちゃんとした世界ができていなくちゃ．

C平　おいおい，なんでWindows 3.0がでてくるんだよ．ソリティアで遊んでるだけなくせに．

P子　だから～，ソリティアみたいなゲームだけやるんなら，別にWindows 3.0でなくてもいいようなもんじゃない．でも，しっかりインストールして使ってみようという気になるしかっりした世界をもっているわけよ．

C平　はいはい，わかりました．Windows 3.0にはまってからP子は強気になっているから……．

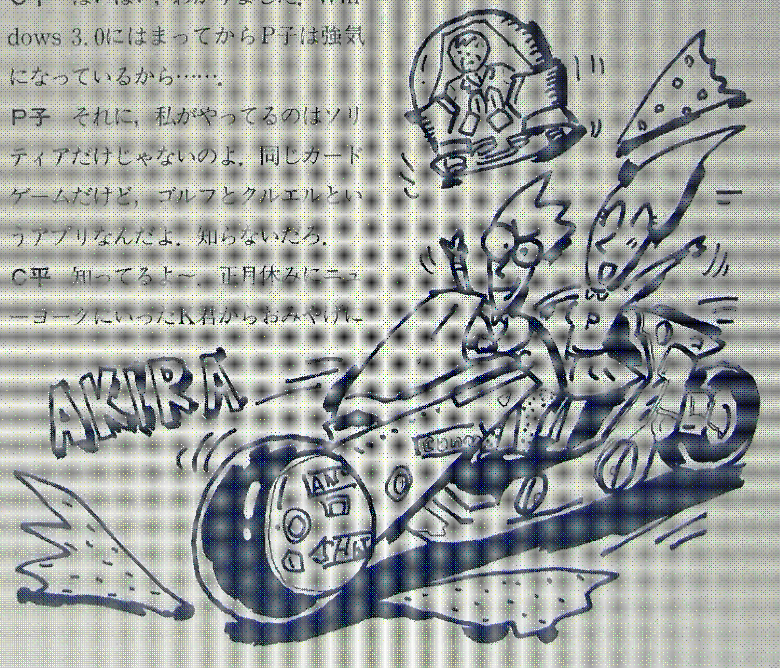
P子　それに，私がやってるのはソリティアだけじゃないのよ．同じカードゲームだけど，ゴルフとクルエルというアプリなんだよ．知らないだろ．

C平　知ってるよ～．正月休みにニューヨークにいったK君からおみやげにもらった「エンターテイメントパック」に入っていることもね．

P子　それなんだけど．エンターテイメントパックというから期待してみてみたんだけど，はじめは，それほどでもないという感じがしたのよ．パッケージには，テトリスが売りだって書いてあるけど，いまさらテトリスでもないなって思ってたの．ところが，やってみたらおもしろいのよ．テトリスじゃなくて，ほかのゲームがね．あれで，数十ドルだっていうから，りっぱなものよ．みんなシンプルなゲームだけど，絶対にはまるから．

なんたってポンポコリン

C平　さっき，「チビまる子ちゃん」の7巻が出たっていってたけど，ぼくが



P子　某ソフトハウスで企画営業を担当している。26歳。
C平　某メーカーのシステムソフトウェア開発部門に所属。25歳。

密やかなたのしみにしていたころはあんなことになるとは思わなかったな。そりゃあ，はじめはアニメになったり，NTTのポスターにでているのを見て「まる子も有名になったものだ」なんて思っていたけど，去年の後半みたいに全国的なブームになってしまうと，単行本買うのもなんだか恥ずかしくなってしまう。そういえば，流行とはそれ自身によって傷つけられるものだと誰かが書いていたな。

P子　何を深刻になってるのよ。なんたって，ポンポコリンだからね。かくて，C平の密やかなたのしみは奪われたというわけ。それで，最近の密やかなたのしみは何になったの？

C平　えー，「リボンマスコットコミック」の「チビまる子」から「花とゆめコミック」の「動物のお医者さん」へと移行したわけです。

P子　あ～。それなら私も知ってる。チョビがでてくるやつでしょ。シベリアンハスキーの。このあいだ家のそばを散歩していたの。般若のもようのあるチョビそっくりのやつが。

C平　はじめは，作者は獣医かと思ったくらいよく取材してある。単行本の巻末に「Making of ……」がついているでしょ。それで，全部調べたのかと感心してしまった。でも最近，できのいいマンガは，すぐにメジャーになってしまうからな。また，傷つけられるかもしれない。

P子　なに，いってんの！　いい年した男が少女マンガを買うこと自体が恥ずかしいのよ。

DATA

■AKIRA　第５巻

大友克洋
講談社

巨大なエネルギーに破壊された後のネオ東京は，さまざまな勢力が割拠する動乱の街だった。そのひとつ「大東京帝国」のリーダーにまつり上げられた鉄雄は，その力を急速に増大させていく。鉄雄の力は天体をコントロールするまでになってしまった。アキラの秘密に迫ろうとする外国勢力，新興宗教，ゲリラ的な少年たちの群れが交錯するネオ東京で決戦のときが近づいている。鉄雄に強烈なライバル意識をもつ金田少年はスタジアムへ突入する。

1,100円

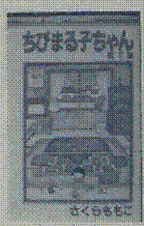
■Microsoft Entertainment Pack for Windows

「上海」のWindows版ともいえる「Taipei」，Windows版「テトリス」などのゲームとスクリーンセイバーの７本を納めたWindows 3.0用のおたのしみパック。アメリカでは年末年始の目玉商品だった。「ゴルフ」と「クルエル」は，「ソリティア」でおなじみになったカードを使ったゲーム。ちなみに「ゴルフ」という名は，ホールインワンのようになかなか上がれないからだとか……。日本語Windows 3.0が発売されると日本でも登場するか？

■ちびまる子ちゃん 第７巻

さくらももこ
リボンマスコットコミック

第６巻から７巻までに「ちびまる子ちゃん」がおかれている状況はまるで違ってしまった。コミックファンの間でのスターが，全国区の有名人になってしまった。第７巻は，文通の話や，熱帯魚をザリガニが食ってしまう話など，すでにアニメ化されたものが多い。このマンガはナレーションをうまく使って，話の展開を支えている。従来，手書きでいれられていたちょっとしたギャグがファンをつかむことはあったが，ここではちゃんとネームになって効果をあげている。



390円

■動物のお医者さん 第１～４巻

佐々木倫子
花とゆめコミック

北海道の高校生ハムテルが地元大学の獣医学部に入るところから物語ははじまる。シベリアンハスキー犬のチョビ，年寄り猫のミケなどの動物たちと，教授，研究生，学生などのからみが不思議なおもしろさを漂わせている。基本的に各話でエピソードを完結させていくが，それぞれが絡み合って登場者のキャラクタを表現しているため，１冊だけ読んだのではおもしろみが半減する。獣医学部の学生に取材し，獣医学教室のディテールを巧みに描き出している。

370～390円

創刊記念プレゼント

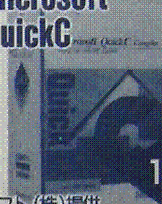
毎月HANDY98を1名様にプレゼント！

2 Turbo C++ Professional

1名

ボーランド・ジャパン(株)提供

3 Microsoft QuickC



1名

マイクロソフト(株)提供

創刊号から創刊第３号で実施！
（商品は月により多少異なります．）

7 dBXL マグカップ

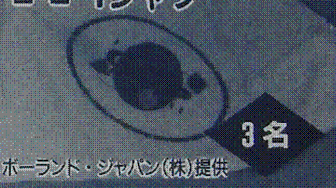
3名

サザンパシフィック(株)提供

1 HANDY98(PC-98HA)

1名

11 OOPツアーTシャツ



3名

ボーランド・ジャパン(株)提供

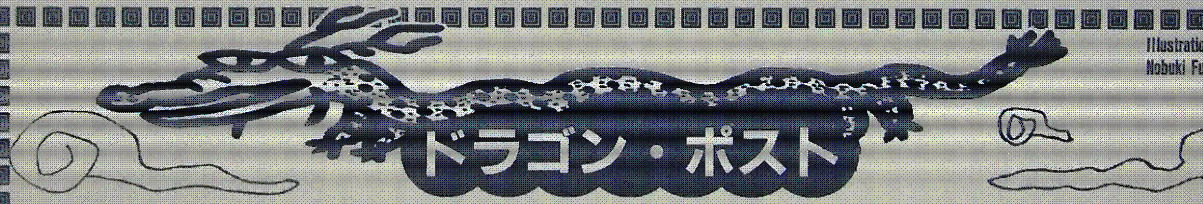


Illustration Nobuki Fu

ドラゴン・ポスト

この欄は，読者からのお便りを中心に構成します．本誌に対するご意見・ご感想などをお待ちしております．鬼より恐い読者からの声ということで，「ドラゴン・ポスト」です．本誌についているアンケートはがきにご意見の欄を設けてありますから，そこに書いてください．ペンネームでも歓迎です．ただし，はがきには本名と住所をしっかり書いてくださいね．掲載誌をお届けします．

★C++学習中

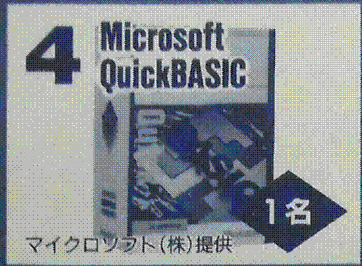
私は，なんとCOBOLのプログラマですが，趣味でCやC++をやります．ただし，C++は学習中なのです．本を読んでも訳がわからんのです．だれがC++はCの拡張だといったのですか？　一応，Cは使えるのに，C++はどうしてもわかりません．確かにコーディングすると，コードは短くなりますが……．まあ，そんなことはおいといて，創刊おめでとうございます．プログラマの目からみて皮肉っぽいところもありましたが，当たっているだけに楽しめました．今月から書籍代が増えそうです．頑張ってください．それではまた来月．（らっぷとっぷ～）

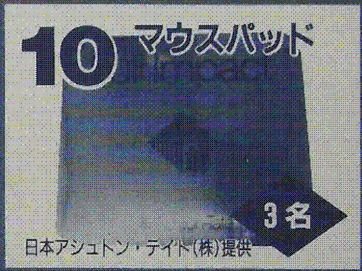
――「それではまた来月」という言葉がうれしい(^_^)．

★汎用機の開発環境にチッソクしそう

私は汎用機のプログラマとか，SEとか呼ばれている“者”をしています．パソコンにもっと近づきたいと思っています．汎用機の時代遅れの開発環境にチッソクしそうです．（ナマケモノ）

――ソースはパソコンのエディタで書いて，大型機などのターゲットマシンでコンパイルして走らせるというプログラマが増えています．その連携の問題もいずれ取り上げてみたいですね．



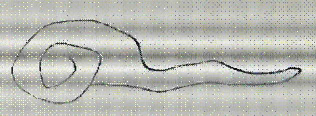


応募は本誌に添付のアンケートはがきをご利用ください（1991年3月17日消印有効）．なお，応募者多数の場合は，抽選になります．当選者発表は発送をもって代えさせていただきます．

★ソフトウェア制作の生産性は？
ソフトウェアを作成する場合，最近ではいいデバッガやエディタが出てきているので，いわゆるMAKING（コーディング，コンパイル，ユニットテスト）の時間は全体の工数の中で5分の1から6分の1程度しかない．後の時間は，打ち合わせやら仕様作成やら，ドキュメント整備で費やされてしまう．この辺の生産性について貴誌で掘り下げて欲しいと思います．（祖倉　哲）
――ソフトウェア制作の工程のこと，そのマネージメントのことなどを具体的に取り上げることができたらと考えています．

★ハサミで切ってしまった
クイズを解くのににらめっこしていたのではわからず，ハサミを使ってしまいました．さて，正解だったでしょうか．（モノズキ）
――やっぱりいましたね．ものずきなかたが．紙を並べるのはなかなかたいへんだったでしょう．このクイズの画面を使ったパズルゲームをソフトウェアサービスしようと準備中ですのでご期待を．

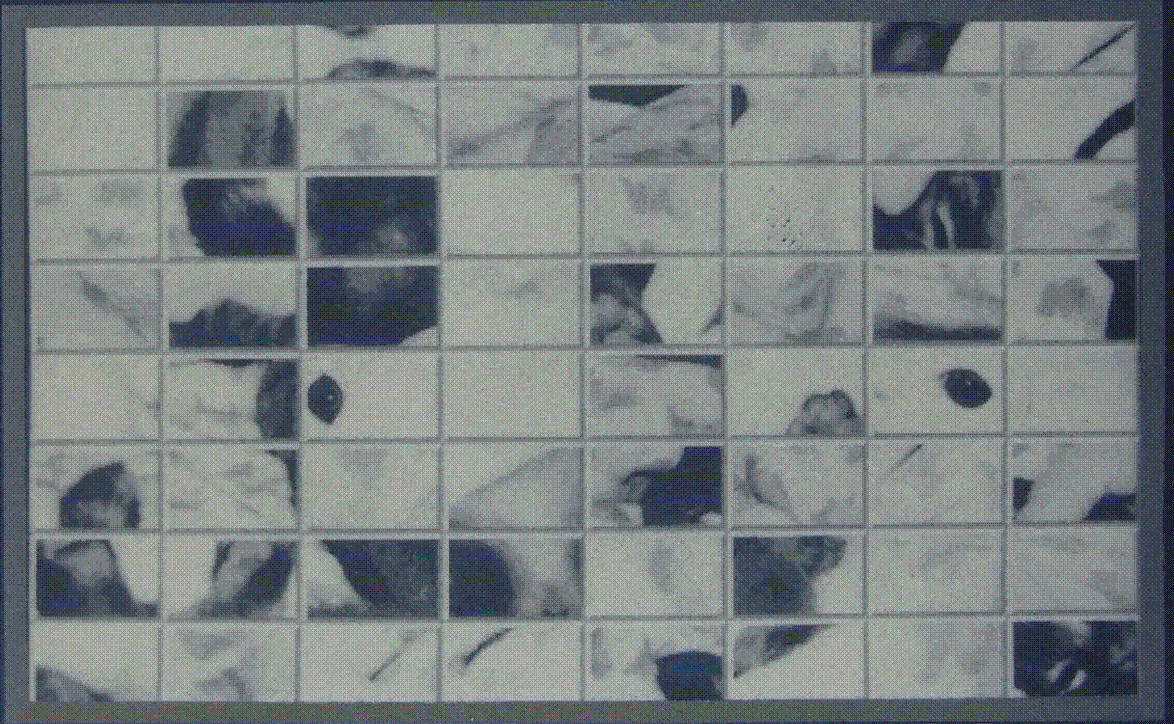
★プログラマとは……
プログラマとは「プログラムをつくる人」のことです．ゆえに，プログラムのつくり方をよく知っていることが当然の前提になります．それで，貴誌に望むのは，プログラムのつくり方よりも，プログラマが直面するさまざまな問題をどのように解決してゆくか，プログラマとしての仕事へのとりくみ方，プログラマとしてのあり方を論じてほしいことです．（佐竹昌道）
――プログラマ像はとても多様で，具体的なイメージを提示しにくいものです．しかし，プログラマの生きた情報を，多少とも紙面に反映させたいというのが本誌創刊時のテーマでもあります．今後もそのテーマのために努力いたします．



クイズ・ランダムスクリーン

ある１枚の画像を64分割し，ランダムに並べ替えてあります．
さて，もとの画像は何を写したものでしょうか．

ヒント：炬燵が好きないきもの



正解者の中から30名様に本誌表紙イラスト（日暮修一画）入り月刊プログラマーズ・ページ特製オリジナルマグカップをさしあげます．

応募方法：本誌に添付のアンケートはがきをご利用ください．

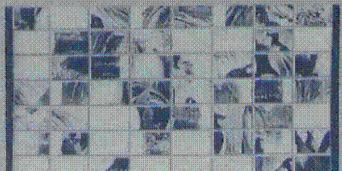
締め切り：1991年３月17日（消印有効）

なお，当選者の発表は発送をもって代えさせていただきます．

＊画像データは，BBSでもサービスしています．

2月号の問題と正解

【問題】



【正解：ペンギン】

月刊プログラマーズ・ページ
プログラムサービスのお知らせ

月刊プログラマーズ・ページでは，本誌に掲載したプログラムのサービスを行っています．
入手方法には，次の2つの方法があります．

1・BBSでのダウンロード

1. 03-5276-1297に電話をかけます．
2. 接続後IDの入力を求めてきますので，GUESTと入力し，リターンキーを押してください．
3. お知らせを表示後，メインメニューのコマンドプロンプトが表示されます．
4. Tと入力するとファイル転送システムに移行します．
5. ダウンロードする目的のプログラムがあるボードの番号を入力します（＊と入力するとボードの一覧が表示されます）．
6. Bと入力するとファイルの一覧が表示されます．
7. ダウンロードするファイルの番号を入力します．
8. ご使用の通信ソフトにあわせて転送プロトコルを番号で選択します（?でプロトコルの一覧が表示されます）．
9. ファイルの転送が終了したら，0と入力し回線を切断します．

2・直接購入

希望メディアサイズ（5インチまたは3.5インチ），プログラムサービス希望の本誌号数をご記入のうえ，現金書留か郵便振替(東京4-154493)，または料金分の切手を同封のうえ，お申し込みください．

●送り先：〒102　東京都千代田区平河町2-4-14
平河町KSビル
株式会社 翔泳社
月刊プログラマーズ・ページ　プログラムサービス係
●価　格：1,500円（消費税，送料，ディスク代込）

・本サービスは入力の手間を省くため，本誌に掲載したソースファイル，実行ファイルを提供するものです．基本的には，掲載時のままでサービスいたしますので，プログラムのなかには，ライブラリファイルのように単独では実行できないものもあります．

・掲載後に発見されたバグなどについては，本誌でお知らせします．また，マニュアルなどは付属していませんので，該当記事をご参照ください．

・プログラムの動作環境，対応機種などについては該当記事でご確認ください．

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

月刊プログラマーズ・ページ4月号

（3月18日発売）

4月号より5インチフロッピーディスク付!!

特集

Cとトコトン付き合おう！

C以外にもいろいろプログラミング言語はあるが，
やはり世の中Cである！
Cと出会い，Cに惚れ込み，Cに迷い，Cに泣き・・・
それでもCと共に生きなければならないプログラマの
ためのC言語大特集！
あなたもCしてみませんか？

月刊プログラマーズ・ページ　1991年3月号
1991年3月1日発行

発行所　株式会社翔泳社
東京都千代田区平河町2-4-14平河町KSビル4F・5F
TEL 03-3263-0447　〒102
発行人　長廻健太郎
編集長　清水 隆
副編集長　鈴木嘉平
編集スタッフ　杉野貴美子＋上田純美礼＋芳山充弘＋勝山まゆみ
デザインスタッフ　佐々木幹夫＋SE（翔泳社デザイン研究所）
広告スタッフ　相澤正夫＋澤本良知＋石崎健治
協力　DOMON＋シグマCC（株）＋小林弘人（図版デザイン）
写植　株式会社東巧社＋株式会社トーヨー・プリント社
印刷　凸版印刷株式会社
定価　980円（本体951円，税29円）

© 1991 SHOEISHA
本書の一部または全部を著作権法の定める範囲をこえ，無断で複写，転載，テープ化，データファイル化することを禁じます。
＊本書の内容に関するご質問は，ご面倒でも必ず文書にて編集部までお問い合わせください。

EDITOR'S PAGE

●日本語Windows 3.0の発売が近いようである．本誌が店頭にでるころには，すでに発表がおこなわれるなり，発売されるなりしているかもしれない．Win3がアメリカで成功を納めたのは，できのいいソフトを受け入れる広範なハードウェアの世界が存在したからである．いくらUNIXマシンのGUIがすばらしいからといって，GUIのためにUNIXマシンが何百万台も売れるマーケットはない．「少ない投資で，たのしいGUI」，それがWin3である．日本語Windows 3.0が「Windowsの春」を迎えられるか，それはWin3を日本のユーザーが「たのしいGUI」と感じてくれるかどうかにかかっているといえそうだ．（TA）

●プログラミング言語の問題で，個人的にもっとも気になっているのが〝オブジェクト指向〟というやつである．ActorやSmalltalk，TurboPascal 5.5やTurbo C＋＋で一応オブジェクト指向というものに触れてみてはいるのだが，どうもいまひとつ摑みきれない感がある．一番重要なことはプログラム設計をオブジェクト指向的に行うことだというのだが，そのへんがどうもねぇ……．しかたなく，今夜も残業明けのゴミプログラミングに励むのであった．(嘉平)

●三島由紀夫が市ヶ谷の自衛隊で割腹自殺を遂げてから20年，なぜかわが家の押入から事件直後のサンデー毎日緊急増刊「三島由紀夫の総括」が現れた．この事件が世間に与えた衝撃の大きさもさることながら，20年間それをとっておいた母のミーハーぶりに感激．20年後に私の子供はプログラマーズページを見て母の若き日を知るのだろうか？（kos）

●本誌創刊号発売の朝，営業担当のT君は，宣伝のため，寒さに震えながら，日本テレビのズームイン朝，通称「オハスタ」前に佇んでいた．ふと，横を見るとサンタクロースの扮装をしたおじさん.「お仕事ですか？」「いいえ，子どもとの約束なんです．クリスマスは家に帰れそうもないので……」．すっかり感激したT君は一歩後ろに下がり，結局，全国宣伝の夢はかなわず．（ま）

●この原稿を書いているのは，米国がバグダットを攻撃した日である．これが第3次世界大戦に絶対発展しないとは，誰も断言できないであろう．そんな時節に，どのプログラミング言語がいいなどと議論している雑誌をつくっている自分に，ふと疑問を感じてしまう．第2次大戦の時も，誰もがこんなふうに自分には何も関係ないと思っているうちに，突然当事者になってしまったのであろう．そう考えつつも，きっと2月18日(P.Pの発行日)も無事むかえているに違いないと，安易に信じきっている自分が怖い．（K）

●新年の目標に「今年こそ衝動買いは止めよう」と誓った．毎年，誓っていることではあるがBSチューナー内蔵ビデオも，MacintoshⅡも，98のHDDも，新車も，スノーボードも，とりあえず購入は延期した．しかし一番使い道のない携帯電話を申し込んでしまった．友人から「また，無駄なモノを……」，「何に使うの……」とか言われ，嬉しさの中から後悔が大きく芽生えようとしている．この調子では今年もお金が貯りそうもない．（よ）

愛読者カードご記入のお願い

本誌をお買い上げいただき、誠にありがとうございます。
お手数ですが、今後の誌面づくりの参考にさせていただきたいと存じますので、
各欄にご記入の上、ご返送くださるようお願い申しあげます。

月刊**プログラマーズ・ページ** READER'S SURVERY CARD　愛読者カード

本誌の号数　　年　　月号

本誌を知った理由
- □1.書店店頭　□2.他社出版物(　　　)
- □3.小社出版物　□4. DM　□5.口コミ

パソコンの所有形態(機種名)
- □1.個人所有　(　　　)
- □2.職場にある(　　　)
- □3.持っていない

よく使用するソフトウェア

よく使用するOS

今月号でおもしろかった記事(いくつでも)

今後取り上げてもらいたい特集のテーマ

貴方が選ぶベスト3

パソコン雑誌	①	②	③
ソフトウェア	①	②	③
パソコン本体	①	②	③

本誌についてのご感想、お気づきの点、その他どのようなことでも結構ですので、ご意見をお聞かせください(「ドラゴン・ポスト＝読者の声」欄に掲載させていただく場合もあります。ペンネームをご希望の方はあわせてご記入ください)。

月刊**プログラマーズ・ページ** 広告資料請求カード

8	会社(学校)名		9	所属(学部)		10	従業員数	

※下記の該当する番号をご記入ください。

11	業種		12	職種		13	パソコン利用状況	

業種　①製造業　②金融・保険・不動産業　③商社・卸・小売業　④建設業　⑤通信・運輸　⑥システム・ソフト　⑦その他サービス業　⑧公務員　⑨その他

職種　①コンピュータ(ハード)技術者　②コンピュータ(ソフト)技術者　③エレクトロニクス技術者　④プログラマ　⑤その他の技術者　⑥専門職　⑦事務・営業職　⑧経営管理職　⑨商工サービス　⑩教職　⑪学生　⑫その他

パソコン利用状況　①会社で利用　②個人で利用　③会社でも個人でも利用　④利用していない

●希望される広告資料　　年　　月号

希望項目 ＼ 広告資料請求No.												
A 購入希望	A	A	A	A	A	A	A	A	A	A	A	A
B 説明希望	B	B	B	B	B	B	B	B	B	B	B	B
C 資料希望	C	C	C	C	C	C	C	C	C	C	C	C

定期購読のご案内

綴じ込みの振替用紙を使って、本誌の定期購読をお申し込みになれます。
1年間12冊＝11,760円（本体11,417円/税343円）
郵送料サービス

創刊記念 HANDY 98 プレゼント

郵便はがき

料金受取人払

麹町局承認

3908

差出有効期間
平成4年11月
30日まで

切手不要

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
マーケティング部 行

月刊プログラマーズ・ページ READER'S SURVEY CARD 愛読者カード

氏名	性別 □1男 □2女	生年月日 19　年　月　日
住所 〒		電話 （　　）　－

職業
□1ソフト技術者 □2ハード技術者 □3技術系会社員
□4事務系会社員 □5営業系会社員 □6教師
□7一般公務員 □8経営管理職 □9理系学生
□10文系学生 □11その他（　　　）

ユーザー区分
□1プロ
□2マニア
□3一般ユーザー

創刊記念プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

郵便はがき

料金受取人払

麹町局承認

3910

差出有効期間
平成4年11月
30日まで

切手不要

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係 行

＊当社で記入

1	SE	2	受付No.	

月刊プログラマーズ・ページ広告資料請求カード

3	会員番号	

＊すでに会員番号をお持ちの方はご記入ください。

4	フリガナ 氏名	姓　名	5	生年月日	19　年　月　日
6	自宅住所	〒　電話（　　）			
7	会社住所	〒　電話（　　）			

資料送付希望先の番号に○印をつけてください。

払込票

通常払込料金加入者負担

口座番号 東京 4 15493

加入者名 株式会社 翔泳社

金額 11760

※ 払込人住所氏名

受付局日附印

備考

切り取らないで郵便局にお出しください。

記載事項を訂正した場合は、その箇所に訂正印を押してください。

払込通知票

通常払込料金加入者負担

口座番号 東京 4 15493

金額 11760 円

加入者名 株式会社 翔泳社

払込み 料金 特殊 円

備考

受付局日附印

※（郵便番号　　　）

TEL（　　　　）

払込人住所氏名

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）

各票の※印欄は、払込人において記載してください。

ト株式会社

ケイツ氏が創業したマイク
フィスとすべての家庭にコ
げ、1974年に活動を開始し
を擁する大企業グループと
。同社が開発したプログラ
ケーション製品が世界各国
準"となっていることは、
日本での企業形態や採用の
にオフィスを構える本社の
た。

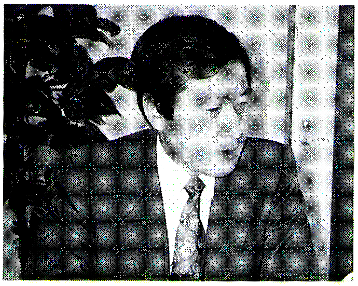
ラム全体の責任をとる、すな
ログラムをかくという仕事のま
を「プログラムマネージャ」と
一方「プロダクトマネージャ」
ういった対象の人にこの製品を
売るにあたってどういう機能
てはいけないか、何が必要で何
かといった判断を下し、プログ
ネージャと相談しながら工程す
管理していく職務である。仕様
るという意味において、営業マー
ングセンスが要求され、プログ
理解している必要があろう。
では、２つの職制はほぼ同人数
本人の希望と能力と適性が、
の道に進むかの指標となる。

比させて適性を見る。将来性という意味で、発想力や企画力も要求される。

二つ目は「人間性」である。前向きであるか、チームワークがうまくとれるか、例えばアメリカへ一人で行ってこいと言われて、内心ビビッてもなんとかやってみようとする積極性があるか、といったパワーの有無を見極める。

人材はまだまだ足りないというのが本音だ。しかし、一人一人の生産性を高

になるか「プロダクトマネージャ」になるかである。



マイクロソフト株式会社　人事・総務部部長　鶴野剛夫 氏

外資系＝英語ペラペラか？

マイクロソフトは、100％外資系の企業である。外資系イコール英語力という条件式は、同社にも成り立っているのだろうか。例えばC言語やアセンブラがバリバリにできても、英語がまるっきりダメという人は採用されないのだろうか。

「私はこんな質問をします。"英語に対

定期購読のご案内

■定期購読は、1年間12冊と限定させていただきます。
■購読料のお支払は、この振替用紙をご利用ください。銀行振込、現金書留によるご送金は、ご遠慮ください。
■振替用紙に、ご住所、お名前をご記入のうえ、郵便局の貯金窓口にてお支払ください。
■小社より領収書の発行はいたしませんので、郵便局にて「郵便振替払込金受領証」をお受取ください。
■お申し込みいただいた方には、弊社より直接発送いたしますので、到着が発売日より遅れることがございますが、ご了承ください。

月刊プログラマーズ・ページ　定期購読直接申し込み書

この欄は、加入者あての通信にお使いください。

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

年　　月号より

年間購読料 11,760円(税・送料込み)

■ご希望の購読開始月号をご記入ください。

■送付先（住所・氏名は正確にご記入ください）

住所	□□□-□□	フリガナ	
		氏名	
		TEL	

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。　（郵　政　省）

郵便はがき

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係 行

2	受付No.	

マーズ・ページ広告資料請求カード

＊すでに会員番号をお持ちの方はご記入ください。

名	5	生年月日	19　年　月　日

綴じ込みの□ 1

創刊記念 HANDY 98 プレゼント

□7一般公務員　□8経営管理職　□9理系学生　□3一般ユーザー
□10文系学生　□11その他(　　　　)

創刊記念プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

6	自宅住所	〒　電話　(　　)	資料送付希望先の番号に○印をつけてください。
7	会社住所	〒　電話　(　　)	

ちょっと気になるこの会社

マイクロソフト株式会社

コンピュータの天才、ビル・ケイツ氏が創業したマイクロソフト（株）は、“すべてのオフィスとすべての家庭にコンピュータを”という目標を掲げ、1974年に活動を開始した。以来16年、全世界で6000人を擁する大企業グループとなり、今なお成長を続けている。同社が開発したプログラム開発用言語やビジネスアプリケーション製品が世界各国で採用され、各分野で“世界標準”となっていることは、周知の事実である。拠点となる日本での企業形態や採用の動向が気になる編集部は、新宿にオフィスを構える本社の人事・総務部長、鶴野氏を訪ねた。

なんと中途採用者が95％

マイクロソフト社の平均年齢は29歳と若い。しかも大半が中途入社だとは聞いていたが**「社員200人中で中途採用者が190人なんです」**という鶴野氏の言葉には、いささか驚いた。通常と逆で、中途採用者に囲まれた新卒者の方に気を遣ってしまうこともあるという。毎月５人～８人の募集に対して応募者は100人近くにものぼる。そんな狭き門をくぐり社員として選ばれる人は、いったいどんな条件を満たしているのだろうか。

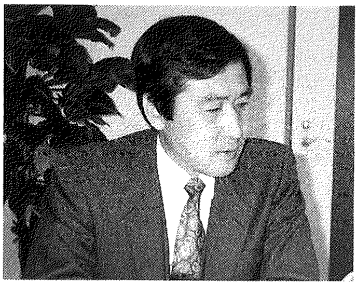
採用にあたって同社が重視するのは２点だ。一つは「技術力」で、Ｃ言語やアセンブラの実務経験と年齢とを対比させて適性を見る。将来性という意味で、発想力や企画力も要求される。二つ目は「人間性」である。前向きであるか、チームワークがうまくとれるか、例えばアメリカへ一人で行ってこいと言われて、内心ビビッてもなんとかやってみようとする積極性があるか、といったパワーの有無を見極める。

人材はまだまだ足りないというのが本音だ。しかし、一人一人の生産性を高く設定しているため、選考基準もなかなか高い。100点満点で入社する人はほとんどいないということだ。

入社後は2つの道を選択

では、入社後の教育システムは、どうなっているのだろうか。

中途採用の場合、入社日の午前中はオリエンテーションで、午後からはもう現場に出る。先輩社員の指導で、マニュアルを読むところから始める人、先輩の補佐的な仕事から入る人、比較的軽いパートを任される人……と、個々のレベルに合ったＯＪＴがなされてゆく。やがて力をつけたエンジニアは、分岐点に立つ。**「プログラムマネージャ」になるか「プロダクトマネージャ」になるかである。**



マイクロソフト株式会社　人事・総務部 部長　鶴野剛夫 氏

プログラム全体の責任をとる、すなわちプログラムをかくという仕事のまとめ役を「プログラムマネージャ」と呼ぶ。一方「プロダクトマネージャ」は、どういった対象の人にこの製品を売るか、売るにあたってどういう機能がなくてはいけないか、何が必要で何が不要かといった判断を下し、プログラムマネージャと相談しながら工程すべてを管理していく職務である。仕様を決めるという意味において、営業マーケティングセンスが要求され、プログラムも理解している必要があろう。

同社では、２つの職制はほぼ同人数である。本人の希望と能力と適性が、どちらの道に進むかの指標となる。

外資系＝英語ペラペラか？

マイクロソフトは、100％外資系の企業である。外資系イコール英語力という条件式は、同社にも成り立っているのだろうか。例えばＣ言語やアセンブラがバリバリにできても、英語がまるっきりダメという人は採用されないのだろうか。

「私はこんな質問をします。“英語に対

してどういうふうに取り組んだことがありますか”“英語は嫌いですか”“英語が喋れたらいいなあと思いますか”“喋れるようにと努力するつもりはありますか”とね。つまり、今できるできないよりも英語に対して前向きであるかどうかを重視するんです」と鶴野氏は答えた。英語を習得する環境はいたるところにあるという。例えば「電子メール」。世界中にいる社員6000人は一人一台以上コンピュータを持っており、1日平均約30通以上の情報を交換する。当然すべて英語である。目を通さねばならない状況にあって、苦手な人は英語の方が得意な日本人やバイリンガルといった周囲のコーチを大いに利用する。また、英会話の通信教育は会社の100％補助で受講させているし、部門長推薦によって、就業時間内に社内で開く英会話スクールへの参加も奨励している。これだけの環境が整っていたら、かなりの英語力が身につくはずだ。しかし逆にできないということは相当のネックになるのではないだろうか。思い切って本音を聞いてみた。

「英語ができない人は、肩身が狭いですか？」

「う～ん、ひらきなおってる人が多いですね。アメリカ人と会議する時は、かなり肩身が狭いですけど。そうですネェ、社員の1／3はバッチリ、1／3はなんとかなる、あとの1／3は困ってるっていう状況ですかね」

鶴野氏は笑っていった。

日米合作のトップレベルの待遇を目指す

外資系の悪いイメージとして、例えば人がいらなくなると解雇あるいは減給という、ドライさだけを受け継いだ面が指摘されるが、マイクロソフトは、日米双方の長所を活かした制度をとっている。日本の“終身雇用制度”や“減給を良しとしないやり方”は社員に安心感を与える。一方、年齢に関係なく能力に応じてポジションや給料がアップするというアメリカ方式は、社員のヤル気を倍増させるものだ。故に社員の定着率はかなり高い。

新入社員が早く会社に馴染めるような体制作りもよくできている。入社後3カ月以内にディナーパーティーを開き、社長らと直接コミュニケーションを図る場を提供する。バラバラに入った同期の仲間意識もそこから生まれる。また、毎月各部で懇親会をやることを奨励しており、実施した部署には補助金を出したりもしている。

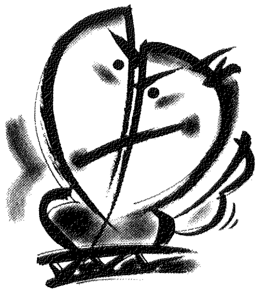
それもこれも創業者ビル・ゲイツ氏の『我々の最も価値のある財産は社員である』というポリシーが浸透しているからに他ならない。ゲイツ氏は日本に大変興味があり前向きだ。これからの技術は日本から出てくるということを、彼は十分知っている。年に数回来日するが、早朝から夜中まで自分のスケジュールが詰まっていないと、イヤな顔をするらしい。自分は必ず役に立つ存在なのだから、日本にいる間はフルに使えというのだ。

●

強大なパワーを持って全世界を駆け回るゲイツ氏を中心としたマイクロソフトは、まさに“世界の最新情報発信基地”としての役割を十分担っている企業と言える。安心して働ける環境と夢や理想への最短ルートを知り尽しているところは、就職・転職希望者にとっては頼もしい限りである。

『この心の叫びを聞いてくれ！』のコーナーを新設します。
このコーナーでは皆様の仕事、職場環境などに対するご不満、お怒りを募集しています。官製ハガキにて下記にお送りください。抽選で毎月10名に粗品を進呈いたします。

東京都千代田区平河町2－4－14　平河町KSビル
〒102　㈱翔泳社　マーケティング部



私たちの責任は、時代とともにますます高度化しています。マイクロソフトが掲げた“すべてのオフィスとすべての家庭にコンピュータを”という言葉は、情報社会の急速な進展とともにその意味をさらに深めています。マイクロソフト㈱は、米国マイクロソフトの日本法人として1986年に設立され、《世界標準》の新たな技術開発拠点として急成長。日本という枠を越えたグローバルな活動を幅広く展開しています。日本発の世界標準が生まれる日も遠くありません。**オフィスは創造力を刺激する最先端のインフラです。**ひとり当たり4m²以上のゆったりしたスペース、全社員に1〜3台以上のターミナル。クリエイティブ・エリアとして、コミュニケーション・エリアとして、国際ネットワークを通じてマイクロソフトの「最新」をデスクからアクセスしながら、能力を存分に発揮できる環境を用意しています。**自由と自主性を核とする、人間中心の組織体です。**フレックス・タイム制の導入、服装もフリー。マイクロソフトには個人の自主性を尊重するカルチャーがしっかりと根づいています。細かい制約にとらわれない企業環境のなか、自分なりのスタイルで充実感を得ることができます。**マイクロソフトは、あなたの自己実現に最良の環境を提供します。**“ソフトウェア・テクノロジーの歴史はそのままマイクロソフトの歴史”とまで言われている環境の中で、あなたが活躍するステージは、世界です。自分自身の目標を実現するのに最適な環境、確立すべきPI（パーソナル・アイデンティティ）や自己実現への可能性を加速させる環境が、ここにはあります。

ここはクリエイティビティの王国です。

Microsoft®

■募集職種

①研究開発ⓐネットワークシステム開発エンジニアⓑオペレーティングシステム開発エンジニアⓒ言語製品開発エンジニアⓓアプリケーション開発エンジニアⓔテクニカル・ライティング・スタッフ②テクニカル・サポート・スタッフ③ソフトウェアの品質管理、製品教育④マーケティング・スタッフ⑤営業スタッフ⑥財務、人事、総務

■募集要項

●資格：22歳〜28歳ぐらいまでの方。●給与：経験、年齢、能力、現行給与を十分考慮のうえ、当社規定により優遇。中途入社によるハンディは一切ありません。●勤務地：原則として東京。職種により長期海外出張あり。●待遇：昇給年1回、賞与年2回。●勤務時間：9：00〜17：30。フレックスタイム制（コアタイムは11：00〜15：00）●休日休暇：完全週休2日制、祝祭日、年次有給休暇、夏期休暇、年末年始休暇、傷病休暇（有給）、慶弔休暇など。●福利厚生：団体定期保険・団体医療保険（全額会社負担）、社員持株制度、クリスマスパーティー（全額会社負担）、スポーツクラブ、通勤手当全額支給、各種社会保険完備、退職金制度、財形貯蓄制度など。

■会社概要

●設立：1986年2月17日●資本金：1億円●売上高：66億円（1990年6月期）●代表取締役社長：古川 享●従業員数：200名●事業内容：パソコン用ソフトウェア（OS、言語、アプリケーション）およびそれに関連する製品の開発・販売●事業所：東京都新宿区西新宿（JR新宿駅より徒歩約10分、西武新宿駅より徒歩約3分）、米国本社/マイクロソフト・コーポレーション（ワシントン州レッドモンド）

■応募方法

希望職種を記入の上、履歴書（写真貼付）および職務経歴書を郵送ください。書類選考のうえ、面接の日程をお知らせします。※応募の秘密は厳守します。※応募書類は返却しません。※入社日は相談に応じます。

連絡先

マイクロソフト株式会社

〒160 東京都新宿区西新宿7-5-25 Kビルディング《S-3係》

人事課：佐藤 TEL 03-3362-8031

あなたの情報アンテナを本づくりへ

編集スタッフ募集

募集要項

職種／雑誌・書籍・マニュアルの編集　**資格**／PCのプログラミング言語・OSについての基礎知識を有し，文章力に自信のある編集経験者あるいは本づくりに関心のあるソフトウェア技術者．　**給与**／平成２年大卒初任給200,000円．経験・年齢・能力を考慮の上，当社規定により優遇．中途採用のハンディは，一切ありません．　**待遇**／昇給年1回．賞与年2回．住宅手当・家族手当・残業手当有り．各種社会保険完備．交通費全額支給．　**休日**／週休2日制．　**休暇**／春・夏・冬それぞれ7～10日間の連休有り．　**勤務**／10：00～18：00　**応募**／履歴書をご郵送下さい．追って詳細をご連絡致します．

会社概要

事業内容●コンピュータ・エレクトロニクス・情報処理関係の書籍・雑誌出版　●大手メーカーのソフトウェアマニュアルの企画・編集制作　●CIをはじめとする各種デザイン事業　**設立**／昭和60年12月19日　**資本金**／2,000万円
売上高／13億5,600万円（平成元年度）

SE

株式会社 翔泳社　〒102 東京都千代田区平河町2-4-14 平河町KSビル4F・5F TEL.03-3263-0447 担当：人事係

Microsoft C 6.0関連プログラム起動オプション

■CL.EXE

構文：CL［options］［filename］... ［libraries link-options］

〈options〉	コンパイラの動作を変更するオプションを指定します．ファイル名とオプションは混在できます．１つのオプションが，コマンドライン上のファイルすべてに適用されることもありますが，直後のファイルだけに適用されることもあります．
/AC	コンパクトメモリモデルの指定
/AH	ヒュージメモリモデルの指定
/AL	ラージメモリモデルの指定
/AM	ミディアムメモリモデルの指定
/AS	スモールメモリモデルの指定
/AT	タイニーメモリモデルの指定
/Au	カスタマイズセグメントの設定
/Aw	カスタマイズセグメントの設定
/B1	ラージモデルのコンパイラパスC1L.EXEを使用
/B2	ラージモデルのコンパイラパスC2L.EXEを使用
/B3	ラージモデルのコンパイラパスC3L.EXEを使用
/batch	バッチファイルやコマンドファイル内でCLの操作を可能に設定
/c	リンクをしないでコンパイル
/C	プリプロセス時のコメントを保存
/D	定数とマクロの定義の指定
/E	プリプロセッサ出力を標準出力へ
/EP	プリプロセッサ出力を標準出力へ
/F	スタックサイズの設定
/Fa	アセンブリリスティングファイルの作成
/Fb	バインドされた実行ファイル名を最初のソースファイル名に設定
/Fc	ソースとオブジェクトの混合リスティングファイルの作成
/Fe	実行ファイル名を最初のソースファイル名に設定
/Fl	オブジェクトリスティングファイルの作成
/Fm	マップファイルの作成
/Fo	オブジェクトファイルをソースファイル名に設定
/FPa	高速代替数値演算パッケージの選択
/FPc	エミュレーション呼び出し数値演算パッケージの選択
/FPc87	8087呼び出し数値演算パッケージの選択
/FPi	インラインエミュレーションを選択
/FPi87	インライン80x87命令を選択
/Fr	ソースブラウザ機能のシンボリック情報を含んだファイルの出力
/FR	拡張ソースブラウザ機能のシンボリック情報を含んだファイルの出力
/Fs	ソースリスティングファイルの作成
/Fx	MASMクロスリファレンスファイル名の指定
/G0	8086の命令セットを有効にする
/G2	80286の命令セットを有効にする
/Gc	Pascal呼び出し規約の指定
/Gd	Ｃ呼び出し規約の指定
/Ge	スタックチェックルーチン呼び出しの生成
/Gi	インクリメンタルコンパイルの指定
/Gm	定数セグメント内への文字列の配置
/Gr	Fastcallの呼び出し規約の指定
/Gs	スタックチェックルーチン呼び出しを生成しない
/Gt	データサイズの限界値の設定
/G〈x〉	コード生成の制御
/H	外部名の長さの制限
/HELP	ヘルプの表示
/I	インクルードファイルを指定された〈directory〉から探す
/J	デフォルトのキャラクタ型の変更
/Jd	DBCS環境ベクタをロードしない
/Lc	互換モードの実行可能ファイルをリンク
/Li	プログラムをインクリメンタルにリンク
/link	リンカコントロールオプション
/Lp	プロテクトモードの実行可能ファイルをリンク
/Lr	リアルモードの実行可能ファイルをリンク
/MA	マクロアセンブラオプション
/MD	Ｃランタイムライブラリを DLLとしてリンク（OS/2）
/ML	Ｃランタイムライブラリを DLLの一部としてスタティックにリンク（OS/2）
/MT	マルチスレッドオプションを可能にする（OS/2）
/ND	セグメントを作成し，dataseg GROUPグループに結合
/NM	ミディアム・ラージ・ヒュージモデルプログラムの TEXTセグメントに命名するモジュール名の設定
/nologo	CL起動時の開始メッセージを表示しないよう指定
/NT	コードセグメント名の設定
/Oa	エイリアスを無視
/Oc	すべてのローカル最適化
/Od	コードの最適化の禁止
/Oe	グローバルレジスタ割り当て
/Og	すべてのグローバル最適化
/Oi	特定の関数を埋め込み関数として生成
/Ol	安全なループ最適化
/On	安全でないループ最適化の禁止
/Op	浮動小数点演算の一貫性の改善
/Or	インラインリターンの禁止
/Os	プログラムサイズ優先の最適化
/Ot	コードの大きさより実行速度を優先
/Ow	関数呼び出しにまたがったエイリアス以外にエイリアスがないと無視する
/Ox	最大限の最適化
/Oz	最大限のループおよびグローバルレジスタ割り当て最適化
/P	プリプロセッサ出力をファイルへ出力
/qc	クイックコンパイルの指定
/Sl	ソースリストの書式オプション（行幅）
/Sp	ソースリストの書式オプション（ページ）
/Ss	ソースリストの書式オプション（サブタイトル）
/St	ソースリストの書式オプション（タイトル）
/Ta	アセンブラソースファイルの指定
/Tc	Ｃソースファイルの指定
/u	定義済みのすべての識別子の定義をオフにする
/U	指定された〈identifier〉の定義をオフにする
/V	バージョン文字列の設定
/W〈n〉	コンパイラが生成するワーニングメッセージの種類の指定
/X	インクルードファイルを環境変数INCLUDEで設定されているディレクトリから探さない
/Za	Microsoft固有の言語拡張の禁止
/Zc	識別子の大文字小文字を区別しない
/Zd	SYMDEB用の行番号の情報を含むオブジェクトファイルの生成
/Ze	Microsoft固有の言語拡張を許可
/Zg	関数宣言の生成
/Zi	CodeView用の行番号の情報を含むオブジェクトファイルの生成
/Zl	デフォルトライブラリ選択の抑制
/Zp	構造体メンバーのパッキング
/Zr	実行時にNULLポインタと範囲外のfarポインタのチェックを有効にする
/Zs	構文のチェック
〈filename〉	処理するソースファイルかオブジェクトファイルの名前，またはリンク時に検索するライブラリの名前を指定します．
〈libraries〉	使用するライブラリ名を指定します．
〈link-options〉	追加するライブラリ，デフォルト以外のライブラリ，ライブラリを検索するパス，およびリンク中に使うリンカオプションを指定します．（ライブラリを指定しないと，CLコマンドは自動的に適切なライブラリを選択します．）

■CV.EXE

構文：CV(P)［options］executablefile［arguments］
（CVPは，プロテクトモード用CodeView）

〈options〉

/B	カラーモニターを使っている場合でも，モノクロで表示する
/C	CodeViewの起動直後に，１つまたは複数のcommand-windowコマンドを実行する
/D	デバッグするプログラム用に大きな領域を取るため，CodeView自身の一部をディスクオーバーレイからを読み込むように指示する
/E	拡張メモリ（EMSメモリ）を使用可能にします．
/F	CodeViewとデバッグ中のアプリケーションの間でビデオページのスワップをしない
/I	8259の割り込みのトラッピングを行うかどうかをCodeViewに指示する
/K	デバッグセッションの記録が制限される
/L	ダイナミックリンクライブラリに対してシンボル情報の検索を行うよう，プロテクトモード用のCodeView(CVP.EXE)に指示する
/M	システムにインストールされているマウスを無視する
/O	OS/2用CodeView（CVP.EXE）にマルチプロセスのデバッグの指定をする

/R	80386プロセッサを使っているマシンで指定すると，80386の４つのデバッグレジスタが使える
/S	画面スワップに設定され，CodeViewが使用しているテキストビデオページと，アプリケーションが使っているテキストビデオページがスワップされる
/TSF	CodeViewのステータスファイル（CURRENT.STS）を読み込むかどうかを指定する
‹executablefile›	デバッグ対象の実行ファイルの名前．このファイル名は省略できません．シンボリックデバッグを行うには，実行ファイルが/Ziと/Odオプションでコンパイルされていなければなりません．リンクを別のステップで行う場合は，/COオプションでリンクします．
‹arguments›	‹executablefile› ファイルに渡すコマンドライン引数．CodeViewを開始した後，RunメニューのSet Runtime Optionsコマンドで，引数を渡すこともできます．

■PWB.EXE　構文：PWB　[options]　[files]

<table>
<tr><td>‹options›</td><td>すべてのオプションは，大文字・小文字の区別があります．オプションなしでPWBを起動すると，最後に編集していたファイルをオープンし，PWBを終了させたときのカーソル位置へジャンプします．プログラムリストは何も設定されません．</td></tr>
<tr><td>/D[S|T|A]...</td><td>以下の文字で指定した初期設定ファイルを使いません．
S　CURRENT.STSを無視する
T　TOOLS.INIを無視する
A　拡張機能のオートロードを行わない
単に/Dと指定すると，PWBは両方の初期設定ファイルを無視します．
注意：/DTスイッチを指定してPWBを起動すると，セッション中に変更されたキー割り当てやエディタ設定は，保存されません．</td></tr>
<tr><td>/P{P|F}　[‹file›]</td><td>Program Listの指定をします．</td></tr>
<tr><td>/PL</td><td>最後に使ったプログラムリストをオープンします．指定できるプログラムリストは，１つだけです．</td></tr>
<tr><td>/e cmd-string</td><td>起動時に実行するコマンドを指定します．‹cmdstr›内にスペースを含めるときは，ダブルクォーテーション(")で囲んでください．この場合，ダブルクォーテーション自身は¥"，円記号自身は¥¥と指定してください．</td></tr>
<tr><td>/r</td><td>読み出し専用モードでPWBを起動します．ファイルの更新はできません．</td></tr>
<tr><td>/m mark</td><td>前回の終了時の状態ではなく，指定したマークの位置から起動します．</td></tr>
<tr><td>‹[/t] files›</td><td>起動時にロードするファイルを指定します．スペースで区切って複数のファイルを指定した場合は１つ目のファイルだけがロードされますが，Exitコマンドまたは関数を実行すると，そのファイルが保存され，次のファイルがロードされます．ファイル名の前に/tを指定すると，PWBはそのファイルをテンポラリファイルとしてロードします．PWBはFilesメニューのファイルヒストリに，テンポラリファイルを含みません．複数のテンポラリファイルを指定するときは，各ファイル名の前に，それぞれ/tを指定してください．注意：コマンドライン上で/t オプションは最後に指定してください．ファイルの指定の後に，他のオプションを続けることはできません．</td></tr>
</table>

■LIB.EXE　構文：LIB inlibrary [options] [commands] [,[listfile] [,[outlibrary]]] [;]

‹inlibrary›	修正する入力ライブラリ．省略できません．
‹options›	LIBの実行方法を指定するオプション．
/HELP	オンラインヘルプを表示する
/I[GNORECASE]	大文字小文字を区別しない
/NOE[XTDICTIONARY]	拡張辞書を作成しない
/NOI[GNORECASE]	大文字小文字を区別する
/NOL[OGO]	LIBの開始メッセージを表示しない
/PA[GESIZE]:n	ライブラリのページサイズを設定する
/?	簡単なヘルプを表示する
‹commands›	ライブラリに対して，モジュールの追加，削除，置換，コピー，移動を行うために使うコマンド．
+	ライブラリにオブジェクトファイルまたはライブラリを追加する
−	ライブラリからモジュールを削除する
−+	ライブラリのモジュールを置き換える
*	ライブラリのモジュールをオブジェクトファイルにコピーする
−*	モジュールを移動する（モジュールをコピーし，削除する）
‹listfile›	クロスリファレンスリストファイルの名前．名前を指定しないとファイルは作成されません．
‹outlibrary›	入力ライブラリから作られる出力ライブラリ．出力ライブラリを指定しないと，入力ライブラリが修正されます．

■HELPMAKE.EXE　構文：HELPMAKE [options] filename

‹options›	/E, /D, /HELPオプションのいずれかを必ず指定してください．
/Ac	ソースファイルで使っている固有の制御文字をcで指定する（MS-Cでは:を使用）
/C	コンテキスト文字列の大文字／小文字を区別する（実行時にヘルプトピックを検索するときは常に大文字／小文字を区別する）
/D	クロスリファレンスと書式情報を含めヘルプファイルを完全にデコードする
/DS	連結されているヘルプデータベースを個々に分割し，元の名前を付ける（ヘルプファイルが１つのデータベースの時はデコードしない）
/DU	デコードするがクロスリファレンスと書式情報は削除する
/En	ヘルプソースファイルからヘルプデータベースを指定の圧縮比でエンコードする
/H	MAKEHELPの構文の要約を表示する
/HELP	MAKEHELPのオンラインヘルプの呼び出す
/Kfilename	単語の区切りとして扱われる文字を設定する（単語の分割をする文字を含む１行のテキストが記述されたファイルを指定する）
/L	HELPMAKEがデコードできないように，生成したファイルをロックする
/Ooutfile	生成する出力ファイルを指定する
/S1	入力するソースファイルがリッチテキストフォーマットのとき指定する
/S2	入力するソースファイルがQuickHelpフォーマットのとき指定する
/S3	入力するソースファイルがアスキー形式のとき指定する
/T	ドットコマンドとアプリケーション固有のコマンドで変換する
/Vn	エンコード時の診断出力のレベルを設定する
/Wn	ヘルプテキストの１行あたりの文字数を設定する
‹filename›	ターゲットとなるファイル（エンコードするソースファイルまたはデコードするヘルプファイル）

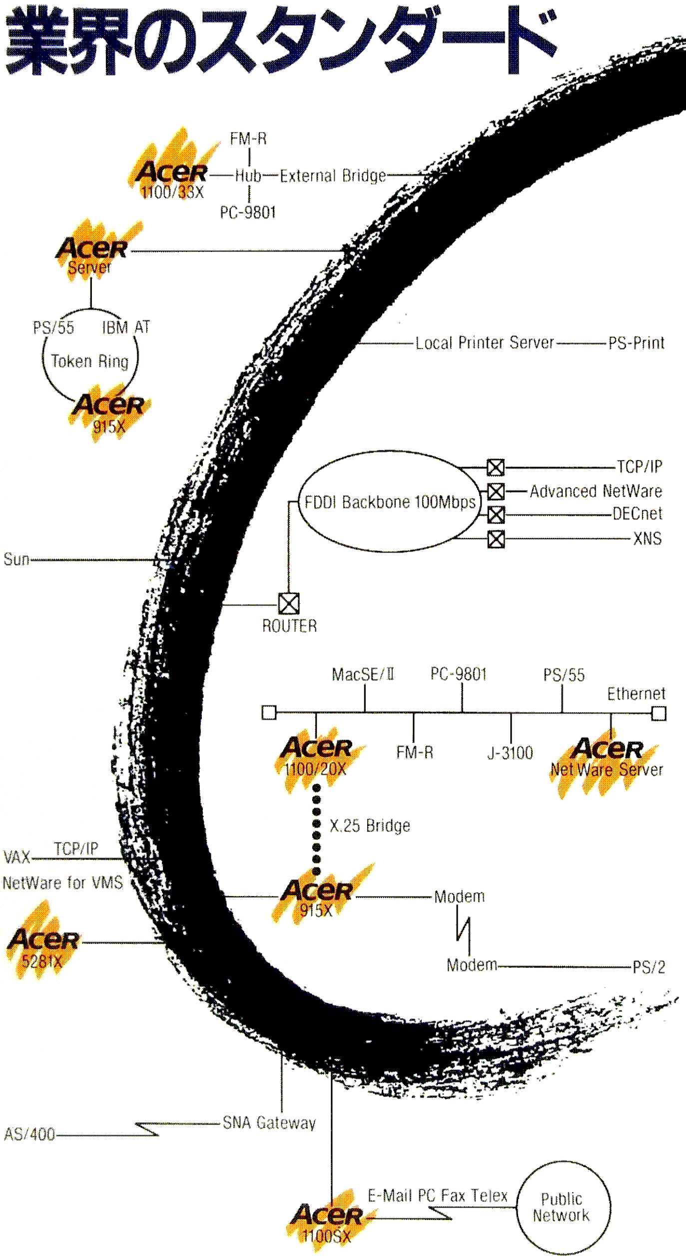
■ILINK.EXE　構文：ILINK [options] projname [modulelist]

‹options›	
/A	前回のリンク処理以降にオブジェクトファイルの変更があったかを調べ，変更されたファイルはインクリメンタルリンクされる
/C	全てのパブリックシンボル名の大文字／小文字を区別する
/E"command"	インクリメンタルリンクに失敗したとき，実行するコマンドを指定する
/HELP	ILINKのオンラインヘルプの呼び出す
/I	インクリメンタルリンクに失敗したとき，リンクを中止する
/NOLOG	起動時の開始メッセージを表示しないようにする
/V	変更のあったモジュールを全て表示する
/X	拡張メモリの使用を禁止する
/?	ILINKの構文の要約を表示する
‹projname›	インクリメンタルリンクを行うプロジェクト名，ファイル名を指定する
‹modulelist›	モジュールリスト名を指定する

NetWare

Acer
The Word for Value

パソコンLANにおける業界のスタンダード



THE ARCHITECTURE EXTENDED-AX

世界のパソコンLAN市場をリード

米国Novell社の"NetWare"は、世界のパソコンLAN市場において60%以上の圧倒的なシェアをもつLAN OSです。
このNetWareに日本語が付加されたことによってACERなどのIBM-PC(AT互換機)はもちろんのことAXマシン、PC-9801、FM-R、PS-55、J-3100などの国産PCへの対応も広がっており、日本語・英語両環境でのネットワーク構築を可能といたしました。

SFT NetWareの特徴

SFT NetWareは、ミニコンやメインフレームでしか提供できなかったさまざまな機能をパソコンベースで実現することができます。他のLANシステムと比較した場合の大きな特徴の一つにシステム・フォルト・トレラント（SFT）機能があげられます。例えばファイルサーバーに2台のハードディスクを装着させることによって"ディスクの二重化"が可能であり、常に同じ情報が2台のハードディスクに格納されるため、万一メインのハードディスクに故障が生じても、サブ側のハードディスクが動作し続け通常業務に影響を与えることはありません。(ディスク・デュープレックス機能)
また、書き込みのたびに、そのセクターを読み出し、セクターに不良があるとわかったら、以後そこを使わないようにする不良セクター対策（Hot Fix機能）やOSにとって重要なディレクトリー・テーブルとファイル・アロケーション・テーブルについては、それぞれ別の場所に書き込んでおき、読み出しの際には、コピーと比較してエラー検査をします。(ディレクトリー構造の二重化、ファイル・アロケーション・テーブルの二重化など…)
以上のようにセキュリティと障害対策が他社製品に比べて非常に強力であることがSFT NetWareの特徴といえます。

●NetWareとTCP/IPの共存が可能

TCP/IPプロトコル、PC/NFS機能によりUNIXマシンおよびSUNなどのワークステーションをサーバーにするシステムをサポート。これによってNetWareとTCP/IPソフトが同時に共存可能になりました。

●D-Link　ポケット・イーサネットアダプター　新発売

J-3100の様なラップトップAT互換機のパラレルポートに接続できるポケット型イーサネットアダプターです。(NetWare対応)

世界の舞台で磨かれた
Acerファイル・サーバー Acer1100/33 (PC/AT互換機)をベース

- i80386(33MHz)CPU,RAM 4MB
- 5.25インチ(1.2MB)FDD×1基、
- 100MB～680MB HDD最大2基 実装可能
- 14インチモノクロモニター、モニターI/Fカード
- 16ビットバス用 Ethrnetカード1枚 標準装備
- フロアスタンド他

*Net WareはNOVELL Inc.の登録商標です。

丸文株式会社

東京都中央区日本橋大伝馬町8-1 丸文ダイヤビル 〒103 第1事業本部 海外営業部C&Cシステムグループ TEL (03)3639-9897(ダイヤルイン)

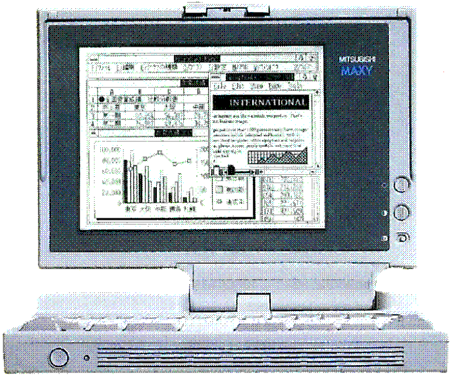
〈資料請求番号P03〉

MITSUBISHI

技術がつくる高度なふれあい SOCIO-TECH

片手に乗った、スーパービジネス。

ライバルに差をつける、32ビットノートパソコン。



スーパービジネスは今、MAXYNOTE386から始まる。

「時代の最先端を切り開くビジネスエリートのために。」スーパービジネスに対応する、MAXYNOTE386の誕生です。A4ファイルサイズのコンパクトボディに、高速32ビットCPU386™SX(16MHz)を搭載。さらにノート初の高速演算プロセッサ387™SXも実装可能(オプション)。広範なビジネスの高速処理を実現します。

着脱式大容量ハードディスクパックで、思いのままに、スーパービジネス。

片手でポンッと出し入れ自在、フロッピー感覚の超小形2.5インチハードディスクパック(40MB/20MB)を内蔵。大容量情報を効率的に、思い通りにコントロールできます。さらに、統合操作環境MS-Windows、統合ソフトMAXYMATEなどを標準装備。ビジネスパワーを高める即戦力ツールとして即座に活用できます。

MAXYSTATIONⅡで、さらに拡がる、スーパービジネス。

本体一体形のコンパクトな拡張ユニットMAXYSTATIONⅡをオプションで用意。PC/ATフルサイズカードが2枚利用可能です。また、100MBハードディスクも内蔵可能。PC/ATの豊富なハードウェア資産を活用した、多彩なネットワーク構築など、デスクトップに迫る豊かな拡張力を提供します。

●高性能CPU386™SX(16MHz)搭載、387™SX(オプション)●RAM標準1MB(最大5MB)●FDD 3.5インチ×1●HDD 2.5インチ、交換自在なハードディスクパック方式(40MB/20MB)●非同期内蔵モデム実装可(オプション)●RS-232C×1(最大2)●PC/ATフルサイズカード2枚実装可(オプション)●パワーONで瞬時に画面を再現するレジューム機能●MS-DOS、MS-Windows、MAXYMATE、MAXYメニューシステムを標準装備

AXdoNext

世界標準機・三菱AXパソコン
MAXYNOTE386

標準価格●A12(HDD 20MBモデル)478,000円(税別)●A14(HDD 40MBモデル)578,000円(税別)〈この他に省スペースデスクトップタイプもあります〉

ニーズから発想する三菱情報通信システム Think&Link

AX THE ARCHITECTURE EXTENDED—AX

MS-DOS, MS-Windowsは米国マイクロソフト社、386™SX、387™SXは米国インテル社、PC/ATは米国IBM社、MAXY, MAXYSTATION, MAXYMATEは三菱電機㈱の登録商標です。MAXYメニューシステムは、㈱エー・ジー・ソフトのAZ-MOS(エージーモス)をベースとしたメニューシステムです。AZ-MOSは㈱エー・ジー・ソフトと㈱アイシーエムが開発したインストール方式「IBFファイル(インストールバッチファイル)システム」を採用しています。
世界標準化を推進するAX。日本のソフトウェアメーカ、ハードウェアメーカ、流通各社によるAX協議会★のもと、OS、キーボード、画面、プリンタ部を規格化。国内・海外の豊富なソフトや周辺機器の共通利用を実現します。★1990年12月現在参加553社

三菱電機株式会社

お問合せは…三菱電機株式会社 〒100 東京都千代田区丸の内2-2-3(三菱電機ビル) 本社情報処理機器事業部計画部(03)3218-3285 本社情報処理機器部(03)3218-3515 本社システム営業部(03)3218-3544 本社情報通信システム計画部(03)3218-3236 本社電機事業部(03)3218-3158 ●札幌(011)212-3739 ●仙台(022)264-5656 ●大宮(048)653-0261 ●横浜(045)212-3521 ●新潟(025)241-7220 ●金沢(0762)52-8821 ●名古屋(052)565-3246 ●大阪(06)347-2421 ●広島(082)248-5330 ●高松(0878)25-0009 ●福岡(092)721-2346

カタログ請求券 AXパソコン プログラマーズ 9102

T4911777703987

〈資料請求番号P04〉

雑誌コード17777-03